UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA INFORMÁTICA

PROYECTO FIN DE CARRERA

DISEÑO Y DESARROLLO DE UNA APLICACIÓN WEB PARA LA GESTIÓN AUTOMÁTICA DE PRÁCTICAS

Autor: Javier Vela García
Tutor: Alberto Núñez Covarrubias

Mayo de 2009

A Raquel

be quiet and drive (far away)

Índice

CAPÍTULO 1:
INTRODUCCIÓN

1 Introducción

1.1 Prefacio

El presente proyecto consiste en normalizar el proceso de entrega y corrección de prácticas académicas en un departamento de la Universidad Carlos III de Madrid. En estas prácticas, dirigidas por personal docente perteneciente al departamento, se proponen un conjunto de ejercicios que deben ser resueltos y entregados por los alumnos.

El contenido de las prácticas normalmente está enfocado a la implementación de aplicaciones software que al final de un periodo de tiempo determinado son corregidas por el personal docente. Este proceso presenta varios inconvenientes:

- Alto volumen de prácticas a corregir.
- No existe un procedimiento normalizado, ya que depende de cada docente responsable de las prácticas.
- No existe un repositorio centralizado en el que almacenar el conjunto de prácticas del departamento.
- Los alumnos no tienen información de la calidad de sus implementaciones software durante el periodo de realización.
- El personal docente no conoce la evolución de los alumnos hasta que no finaliza el periodo de entrega.

Por tanto, este proyecto resolverá en la medida de lo posible el conjunto de inconvenientes previamente detallado, lo que exige el diseño e implementación de un sistema que gestione la situación de las prácticas docentes del departamento.

1.2 Objetivos

Como se ha concluido en el punto anterior, el objetivo principal del presente proyecto es

diseñar e implementar una solución viable que permita de forma automática la entrega y evaluación de una serie de prácticas académicas de un conjunto de alumnos.

Dicha solución persigue los siguientes fines:

- Acometer el estudio y comprensión del papel de la fase de pruebas en el ciclo de vida del software, ya que se considera una fase primordial que conlleva la consecución de aplicaciones software de mayor calidad. En este contexto, esta calidad estará dotada de un alto componente pedagógico, ya que el fin primordial es que los alumnos cuenten con una herramienta que les haga incrementar sus resultados en sus ejercicios prácticos.
- Lograr un alto grado de automatización y control en la gestión del proceso.
- Contar con un repositorio centralizado en el que se puedan almacenar las soluciones propuestas por los alumnos.
- Permitir que los propios alumnos entreguen de forma automática los ejercicios a través del sistema a implantar.
- Lograr que durante el periodo de vigencia de las prácticas los alumnos puedan conocer la calidad de los ejercicios que entregan para que así puedan mejorar en todo momento.
- Facilitar al personal docente la evolución de los alumnos en el conjunto de prácticas del departamento.

1.3 Alcance

Según los objetivos anteriormente descritos, en el presente trabajo pretende llevar a cabo las siguientes tareas:

- Recopilación y análisis del conjunto total de requisitos de la solución a implementar.
- Modelado del sistema en el plano conceptual.
- Estudio de las distintas soluciones tecnológicas para conformar una arquitectura que defina el sistema.
- Codificación e implementación de las distintas partes de la arquitectura propuesta.
- Integración del sistema con elementos externos al mismo (servicios web) que se van a ocupar de la corrección de los ejercicios. Hay que señalar que la implementación de

estos elementos está fuera del alcance de este proyecto.

- Generación de la documentación asociada al sistema.

1.4 Estructura del documento

La organización a la que obedece este documento es la siguiente:

- Introducción: En este punto se describe el alcance del presente proyecto, detallando los objetivos principales que persigue.
- Estado de la cuestión: Este capítulo describe el contexto en el que se enmarca el presente proyecto, que no es otro que el ámbito de las pruebas y correcciones automáticas de aplicaciones software.
- Análisis: Este capítulo describe y plantea de forma completa el problema que resuelve la aplicación. Para ello se detallan el conjunto de requisitos que definen el comportamiento de la aplicación y se define el modelado del sistema.
- Diseño: En este apartado se describe de qué manera se va a acometer la solución del problema. Para ello se hace previamente un estudio del conjunto de tecnologías susceptibles de ser empleadas en el trabajo y se define el sistema final a implementar.
- Implementación: Este capítulo describe los aspectos más relevantes del proceso de codificación de la aplicación.
- Manual de usuario: Aquí se incluye la documentación dirigida a los distintos usuarios de la aplicación. Alberga la base de conocimiento necesaria para usar la aplicación de forma correcta.
- Configuración y puesta en marcha: Este capítulo detalla el conjunto de pasos a seguir para configurar, desplegar y poner el sistema en un entorno de producción.
- Gestión del proyecto: En este apartado se incluye la planificación seguida para la realización del trabajo. Asimismo se detalla el conjunto de elementos software, hardware y humanos necesarios para ejecutar el proyecto. También se incluye una valoración económica del proyecto.
- Conclusiones: En este apartado se reflejan las conclusiones a las que se ha llegado después de la realización del trabajo.
- Líneas futuras: Capítulo en el que se propone un conjunto de posibles mejoras y líneas

de trabajo sobre el sistema construido.

- Bibliografía: Contiene el conjunto de referencias bibliográficas y recursos que han ayudado en los distintos aspectos de la elaboración del proyecto.
- Anexo: Contiene el conjunto de acrónimos utilizados en el presente documento.

CAPÍTULO 2:
ESTADO DE LA CUESTIÓN

2 Estado de la cuestión

2.1 El papel de la fase de pruebas en el proceso de desarrollo software

La fase de pruebas en el contexto del proceso de desarrollo software es el proceso utilizado para medir la calidad del software desarrollado. Normalmente, la calidad es entendida en términos de corrección, completitud y seguridad. Sin embargo, también puede incluir otro tipo de requisitos técnicos como capacidad, fiabilidad, eficiencia, portabilidad, mantenibilidad, compatibilidad y usabilidad.

La fase de pruebas no solo implica la ejecución de las aplicaciones con el fin de encontrar errores. Se trata de emplear distintas técnicas que ayuden a incrementar la calidad del software, ya sea para ofrecer un mejor producto al cliente o como método de corrección y aprendizaje. Por tanto, hemos de remarcar que nos encontramos ante una fase extremadamente importante que nos va a permitir marcar diferencias con un producto determinado.

2.2 Definición formal de "Corrección automática de software"

Formalmente, podemos definir la corrección automática de software como la ejecución del proceso de prueba y depuración de cualquier producto software acometido por una entidad en el que obtengamos unos resultados de forma automática.

Para llegar a este proceso automático es necesario que esté definido un proceso de corrección manual de software en la entidad que se esté considerando. Esto incluye:

- Un conjunto de detallado de casos de uso en los que se incluya los resultados esperados.
- Un entorno de pruebas que incluya una base de datos de prueba que pueda ser restaurada a un estado determinado cada vez que se quiera ejecutar un caso de uso.

Estas dos premisas anteriores son muy importantes a la hora de que una herramienta

automática de corrección de software sea efectiva. De todo esto se deduce que la forma correcta para acometer un proceso de prueba y corrección de software es formalizar previamente y de manera adecuada un proceso de pruebas.

2.3 Metodologías viables

En este punto se detallarán las distintas técnicas y aproximaciones utilizadas para la prueba y corrección de programas, englobadas en dos grupos principales: pruebas de caja negra y pruebas de caja blanca.

2.3.1 Pruebas de caja negra

Las pruebas de caja negra comprenden todo tipo de tests que se ejecuten contra una aplicación software sin conocimiento de cómo trabaja a nivel interno el módulo específico a auditar. Cuando esto se aplica a un producto software, el personal encargado de ejecutar las pruebas únicamente conoce las entradas que debe introducir y las salidas que debe esperar.

Por tanto, puede decirse que este tipo de pruebas están enfocadas a los requisitos funcionales del programa, ya que no se requiere de ningún conocimiento añadido para ejecutarlas. De esta forma, el equipo de pruebas y el de desarrollo pueden ser totalmente independientes, lo que permite que el plan de pruebas pueda ser desarrollado una vez que este definido el conjunto de requisitos de la aplicación.

A continuación se detallan las ventajas de este tipo de pruebas:

- Más efectividad en aplicaciones que impliquen gran cantidad de líneas de código que las pruebas de caja blanca (ver apartado 2.3.2).
- El personal de pruebas no necesita ningún conocimiento del lenguaje de programación en el que está codificado el producto.
- El probador y el desarrollador son independientes el uno del otro.
- Las pruebas se realizan desde el punto de vista del usuario.
- Ayudan a encontrar inconsistencias y elementos ambiguos en las especificaciones.
- Los casos de pruebas son diseñados una vez que se termina de definir los requisitos.

Por contra, presentas las siguientes desventajas:

- Sólo es posible probar un número determinado de entradas, ya que abordar todas las posibles combinaciones es algo imposible.
- Sin unas especificaciones claras resulta complicado diseñar los casos de pruebas.
- Puede haber repeticiones innecesarias de los casos de prueba si no se informa al probador de los casos de prueba que ha ejecutado el desarrollador.
- Pueden existir caminos de la aplicación que no hayan sido probados.
- No pueden ser dirigidas a segmentos de código especialmente complejos, ya que es algo opaco al equipo de pruebas.

A continuación se detallan las distintas técnicas de pruebas de caja negra.

Pruebas funcionales

Este tipo de pruebas están enfocadas al conjunto de los requisitos funcionales y su finalidad principal es comprobar que la aplicación se comporta de la manera esperada. Aunque estas pruebas se suelen realizar hacia el final del ciclo de desarrollo, es muy recomendable comenzarlas tan pronto como sea posible para poder comprobar el comportamiento de los distintos módulos y unidades que componen la aplicación.

Las pruebas funcionales se ocupan de evaluar cómo de bien el sistema a auditar ejecuta una determinada función en el contexto en el que se le supone, incluyendo comandos de usuario, manipulación de datos, búsquedas y procesos de negocio.

Pruebas de estrés

Consisten en probar la aplicación utilizando parámetros pesados tales como valores numéricos complejos, entradas de la aplicación más grandes de lo habitual o grandes consultas a la base de datos, cuyo fin consiste en encontrar el punto máximo de estrés que la aplicación puede soportar. Este tipo de pruebas tienen que ver con la calidad de la aplicación en el entorno en el que ésta se ejecuta. La idea es crear un entorno que exija más de la aplicación que bajo circunstancias

normales.

Un entorno de test de este tipo se establece definiendo distintos puntos de prueba. En cada punto se cuenta con un script que se ejecuta contra el sistema. Estos scripts normalmente están basados en pruebas de regresión. Paulatinamente se van añadiendo más puntos de pruebas que probarán la aplicación hasta que el sistema incurra en algún fallo.

Es común encontrar los siguientes errores en las pruebas de estrés:

- Condiciones de carrera: son conflictos entre dos o más tests.
- Pérdidas de memoria: ocurren cuando una prueba reserva una determinada memoria y no la libera una vez acabada la ejecución.

Pruebas de carga

Consisten en probar la aplicación contra cargas pesadas de trabajo para encontrar el punto en el que el servicio que ofrece la aplicación se degrada o deja de cumplir su cometido. Este tipo de pruebas operan en un nivel de carga predefinido, normalmente el más alto que admite la aplicación. Sin embargo, el objetivo no es provocar un fallo en la aplicación, sino mantener el sistema funcionando con una alta carga de trabajo.

En este tipo de tests cobra gran importancia el contar con una base de datos de casos de prueba bastante extensa.

Pruebas a medida

Este tipo de pruebas se realizan al margen de cualquier tipo de plan de pruebas, ya que su objetivo principal es ayudar al personal integrante del equipo de pruebas a comprender y descubrir el funcionamiento de la aplicación.

Su objetivo consiste en que el personal de pruebas opere con total libertad a la hora de utilizar la aplicación, de esta forma pueden explorarse vías de ejecución que no hayan sido contempladas en el plan de pruebas.

Pruebas de usabilidad

Este tipo de tests suelen ejecutarse si el interfaz de la aplicación está en alta consideración y existe uno específico para cada uno de los tipos de usuarios. Por ello, en estas pruebas se trabaja de forma directa e indirecta con el usuario final para comprobar de qué manera perciben el software y cómo interactúan con él.

Pruebas de humo

Este tipo de pruebas también son denominadas pruebas de cordura y su objetivo principal es asegurarse que la aplicación está funcionando correctamente en un nivel determinado. Básicamente consiste en asegurarse que la aplicación cumple sus requerimientos básicos, apoyándose en la comparación del software como si fuera una máquina en relación al humo: si la máquina emite humo es que funciona.

Pruebas de recuperación

Está orientadas para ver cómo de rápido y cómo de bien la aplicación se recupera frente a un fallo crítico, un fallo de hardware u otros problemas catastróficos.

Pruebas de volumen

Estas pruebas están orientadas a comprobar la eficiencia de la aplicación, es decir, cómo de bien gestiona la aplicación un determinado proceso. Para poder ejecutarse, es necesario contar con una cantidad considerable de datos para comprobar las limitaciones de la aplicación en este tipo de aspectos.

Pruebas de escenario

Consisten en crear una combinación de pruebas que sean equivalentes a un escenario real en el que se vaya a encontrar la aplicación.

Pruebas de regresión

Este tipo de pruebas se centran en volver a ejecutar una batería de pruebas previamente realizada después de haber hecho un cambio significativo en el sistema. Tradicionalmente, este tipo de tests suelen ser automáticos y pretenden evitar los posibles puntos de fallo que puedan ser introducidos por el código que repara otros fallos previos.

Las distintas técnicas en este tipo de pruebas difieren en función del elemento en el que se concentran:

- Regresión de fallos: se vuelve a probar un fallo que en teoría ya ha sido solucionado.
- Regresión general y funcional: se prueba si en general el producto sigue cumpliendo con sus especificaciones generales, siempre después de haber incluido una parte nueva de código.
- Portabilidad: se vuelve a probar la aplicación cuando se migra de plataforma.
- Configuración: se vuelven a ejecutar las pruebas cuando se haya introducido un elemento nuevo en el sistema. Por ejemplo, puede ser una nueva versión de una libraría, nueva versión del sistema operativo o un nuevo dispositivo hardware.
- Localización: la aplicación se porta a otro lenguaje, por lo que se vuelven a ejecutar las pruebas.

Pruebas de aceptación del usuario

En este tipo de pruebas es el usuario final el que interactúa con el sistema. Esto está enfocado a ver si el producto final satisface las expectativas del cliente y ofrece el servicio que se espera de él.

Pruebas alfa

En estos tests, los usuarios finales son invitados al centro de desarrollo para que prueben el producto, tomando nota los desarrolladores de posibles comportamientos a contemplar en función del tipo de entradas generadas por los usuarios. Normalmente dichos usuarios trabajarán con un prototipo de la aplicación.

Pruebas beta

Consiste en distribuir una versión beta o preliminar del producto software a un conjunto reducido de usuarios para que puedan probarlo en sus entornos de trabajo e ir notificando periódicamente aquellos aspectos en los que han encontrado algún punto de fallo.

2.3.2 Pruebas de caja blanca

Las pruebas de caja blanca están enfocadas principalmente hacia la lógica interna y la estructura del código de la aplicación. Esto implica que debe existir un conocimiento suficiente de las tecnologías empleadas para desarrollar el software.

Las ventajas que comportan este tipo de pruebas son:

- Al tener que contar con un conocimiento determinado del código, es muy fácil encontrar qué tipo de entradas ayudan a probar la aplicación de forma efectiva.
- Ayuda en la optimización del código.
- Se evitan elementos innecesarios en las fuentes que pueden dar lugar a errores.

Por contra, presentan las siguientes desventajas:

- Siendo el conocimiento de la tecnología un requisito fundamental, se incrementa el coste de las pruebas, ya que se necesita personal cualificado.
- Es imposible probar cada línea de código y ver si puede incurrir en algún fallo del sistema.

A continuación se detallan distintas técnicas de pruebas de caja blanca:

Pruebas unitarias

El desarrollador ejecuta un conjunto de pruebas determinadas para comprobar que un módulo o unidad integrante de la aplicación está funcionando de forma correcta. El ámbito que comprenden este tipo de pruebas depende del desarrollador y de la arquitectura bajo la que se

encuentre el sistema.

Análisis estático

Consiste en analizar el código fuente con el fin de detectar cualquier posible defecto que haga incurrir en algún fallo.

Análisis dinámico

Este tipo de pruebas consisten a grandes rasgos en ejecutar el código y analizar las salidas que genera. Son el siguiente paso a dar después de un análisis estático del código.

Cobertura de sentencias

Estos tests están enfocados de tal forma que toda sentencia que comprende la aplicación sea ejecutada al menos una vez. Se trata de una aproximación completa al funcionamiento de la aplicación.

Pruebas de seguridad

Están enfocadas a comprobar cómo de bien el sistema está protegido frente a distintas amenazas. Se considerará que el sistema es seguro si reúne las siguientes características:

- Integridad: La información sólo puede ser modificada por quien está autorizado y de manera controlada.
- Confidencialidad: La información debe ser sólo legible para los usuarios autorizados.
- Disponibilidad: Debe ser disponible cuando se necesita.
- Irrefutabilidad: El uso y/o modificación de la información por parte del usuario debe ser irrefutable, es decir, que el usuario no pueda negar dicha acción.

Pruebas de mutación

En este tipo de pruebas los desarrolladores prueban una parte concreta del sistema después de haber ejecutado un cambio determinado en él.

2.4 Aplicaciones

En este punto se enumera un conjunto de soluciones software enfocadas a las pruebas y corrección de programas y agrupadas según el tipo de pruebas que ejecutan.

2.4.1 Aplicaciones enfocadas a las pruebas del código fuente

AdaTEST 95

Producto desarrollado por IPL y enfocado a las aplicaciones codificadas en lenguaje Ada. Sus características principales son:

- Pruebas unitarias y de integración.
- Soporte de todas las versiones de Ada.
- Asistente gráfico orientado a preparar las pruebas.
- Contempla la orientación a objetos.
- Análisis estático.

AQtime

Desarrollado por AutomatedQA, es un conjunto de herramientas que permite depurar la gestión de recursos de una aplicación. Sus características:

- Soporta las versiones 1.0, 1.1, 2.0, 3.0 y 3.5 de la plataforma .NET.
- Trabaja con los compiladores de Microsoft, Borland, Intel, Compaq y GNU.
- Puede trabajar de forma integrada con los IDEs Microsoft Visual Studio, Borland Developer Studio y CodeGear RAD Studio.

devAdvantage

Solución de código abierto de la empresa Anticipating Minds. Aspectos reseñables:

- Es código abierto.
- Trabaja con la plataforma .NET.
- Es capaz de arreglar los errores.

CMTJava

Producto de la compañía finlandesa TestWell orientado al lenguaje de programación Java. Características:

- Gran capacidad de proceso.
- Generación extensa de informes en distintos formatos.
- Multiplataforma.
- Cálculos de la complejidad del código.
- Cálculos de la mantenibilidad del código

2.4.2 Aplicaciones enfocadas a pruebas funcionales

.TEST

Producto propietario de la compañía Parasoft orientado a la plataforma de desarrollo .NET. Entre sus características se puede destacar:

- Análisis estático del código de acuerdo con los parámetros establecidos por el usuario.
- Simulación de líneas de ejecución de código
- Soporte integral para pruebas de regresión.
- Total integración con Visual Studio .NET

CifraTest

Solución de la la compañía Tevron. CifraTest es un producto enfocado principalmente a las prueba funcionales automáticas:

- Según sus desarrolladores soporta la mayor parte de aplicaciones.
- Arquitectura abierta.
- Fácil integración con otras aplicaciones.
- Gran facilidad de uso.

Peta

Producto desarrollado por la compañía Verit Informationssysteme. Es una plataforma para ejecutar pruebas funcionales automatizadas en entornos distribuidos:

- Permite realizar pruebas de integración, carga, estrés y regresión.
- Entorno web.
- Multiplataforma.
- Pruebas automatizadas.
- Alto grado de configuración.

CAPÍTULO 3:

ANÁLISIS

3 Análisis

3.1 Contexto

En este apartado se ofrece una visión global del escenario en el que se enmarca el presente proyecto, hecho que pretende ser un punto de partida para el proceso de análisis de la aplicación software.

El entorno en el que se enmarca este proyecto es un entorno académico situado en un departamento de la Universidad Carlos III de Madrid. En dicho departamento se imparten un conjunto de asignaturas. En varias de ellas existen lo que se denominamos prácticas docentes enfocadas a la implementación de aplicaciones software por parte de los alumnos. Fundamentalmente, son ejercicios propuestos por el docente en los que los alumnos codifican programas relacionados con la asignatura en cuestión.

Los alumnos matriculados en las asignaturas deben entregar las soluciones propuestas de los ejercicios, mientras que éstos deben ser corregidos y calificados por el personal docente del departamento.

Dado el alto volumen de prácticas docentes que pueden existir en un conjunto de asignaturas, surge la necesidad de contar con un sistema que permita un alto grado de control, gestión, automatización y corrección de las prácticas con un mínimo de esfuerzo. Por tanto se persigue contar con un procedimiento que permita definir una serie de asignaturas, prácticas y sus correspondientes entregables, así como el conjunto de reglas asociadas que determinen:

- El año académico y cuatrimestre asociados.
- El rango de fechas en las que se realiza y entrega la práctica.
- El método de corrección a utilizar.

Visto todo lo anterior, se infiere que la solución viable a esta circunstancia es el diseño e implementación de una aplicación software que permita una gestión limpia y eficaz de este proceso.

Por otra parte, se necesita que el sistema permita que los alumnos que tienen que realizar las prácticas propuestas vean una clara evolución sobre las soluciones que implementan. Esto es que el alumno pueda obtener del sistema una calificación provisional de la codificación del ejercicio entregado. Sin embargo, se identifica la necesidad de que el alumno pueda ver una evolución de la calidad de su ejercicio, por lo que para ello el sistema debe ofrecerle la relación de las notas del ejercicio asociadas a sus correspondientes entregas. Análogamente, el personal docente debe ser capaz de ver esta evolución.

Por último, cabe señalar que el objeto de este sistema no es implementar los algoritmos correctores de los ejercicios. Se supone que son elementos externos que interactuarían con la aplicación para proporcionar las distintas correcciones. Una posible implementación consistirá en un servicio web que implemente el método corrector al que se puedan mandar ficheros y obtener el correspondiente resultado.

3.2 Usuarios

En este punto se identifican y se definen los usuarios que van a interactuar con nuestro sistema de corrección automática de prácticas docentes.

3.2.1 Administrador

El administrador, típicamente un docente, es el actor encargado de gestionar todos los aspectos del sistema. Esto incluye:

- Alta y baja de asignaturas.
- Definición y borrado de grupos de alumnos.
- Alta y baja de alumnos.
- Alta y baja de prácticas con sus entregables.
- Fijado de la hora de corrección de los entregables.
- Obtención de datos sobre las correcciones hechas.

Cabe señalar que todos los procesos de alta y baja implican una actualización

correspondiente en la base de datos utilizada por la aplicación.

3.2.2 Alumno

Es la persona matriculada en la asignatura que va a realizar los ejercicios propuestos para cada una de las prácticas asociadas a su grupo de matrícula. Para ello entregará un conjunto de ficheros que el sistema almacenará para posteriormente corregirlos.

También es capaz de obtener las notas de las entregas que ha realizado.

3.3 Requisitos

En este apartado se definen los requisitos de la aplicación, es decir, las especificaciones que debe cumplir el sistema a implementar. Los requisitos se estructuran en dos tipos:

- Funcionales: características de funcionamiento de la aplicación.
- Interfaz de usuario: características relacionadas con la accesibilidad y el aspecto visual de la aplicación.

3.3.1 Requisitos funcionales

A continuación se definen los requisitos funcionales del sistema. Para una identificación correcta de los mismos se codifican con el acrónimo RF (Requisito Funcional) más un número correlativo.

RF01

Se requiere que todos los actores que interactúan con el sistema lo hagan a través de un navegador web[1].

RF02

Tanto los alumnos como el administrador deben contar con un identificador de usuario único a la hora de interactuar con la aplicación. A este identificador irá asociada una contraseña. La introducción de estos dos elementos será la condición para que el usuario (alumno o administrador)

1Un navegador web es un elemento software que contienen la mayoría de sistemas operativos, por lo que su facilidad de uso y alta penetración dentro de la comunidad universitaria (docentes y alumnos) hacen de él el cliente idóneo para interactuar con la aplicación.

abra una sesión válida en el sistema que le permita ejecutar distintas operaciones.

El identificador y la contraseña serán almacenadas de manera persistente en el sistema. Se establece que la contraseña no sea guardada en claro, sino que se utilice una función resumen (por ejemplo MD5) para asegurar su confidencialidad.

RF03

Los usuarios deben poder cambiar la contraseña con la que acceden al sistema. Para ello deben introducir un número determinado de veces (dos) el nuevo valor de la contraseña . De esta forma nos aseguramos que el usuario no comente un error que le lleve a contar con una contraseña establecida por error.

RF04

El sistema debe permitir la gestión de asignaturas por parte del administrador. El conjunto de operaciones permitidas serán:

- Alta de asignaturas: el dato a introducir será la denominación de las asignaturas.
- Baja de asignaturas: posibilidad de eliminar cualquier asignatura del sistema seleccionando el elemento de una lista. En esta operación se debe controlar que no haya un grupo definido para esa asignatura, en cuyo caso no debe permitirse la baja.

RF05

El administrador debe ser capaz de dar de alta y borrar grupos. Un grupo debe definirse con la siguiente información:

- Nombre: descripción identificativa del grupo.
- Asignatura: asignatura previamente dada de alta en el sistema sobre la que se va a definir el grupo.
- Año: año académico.
- Cuatrimestre: periodo académico

RF06

Debe implementarse una opción que permita asignar un conjunto de alumnos a un grupo previamente creado en el sistema.

El conjunto de alumnos matriculados en una asignatura determinada se obtendrá a partir de los listados oficiales de matriculados por cada asignatura. Actualmente esto se realiza a través de una aplicación web (http://www3.uc3m.es/grupos) que proporciona un listado en formato excel para cada una de las asignaturas y grupos con la siguiente información:

- NIU
- DNI
- Nombre
- Apellidos

Por tanto, el sistema a implementar debe tomar como entrada los listados y asignar al grupo en cuestión el conjunto de alumnos contenidos en el fichero excel. Estos ficheros son generados a diario en función del estado del proceso de matrícula, por lo que en determinadas fechas variarán su contenido. De esta característica se desprende que la aplicación debe poder cargar en un grupo un mismo fichero de matriculados cuantas veces sea necesario. El proceso deberá asociar únicamente los alumnos que no se encuentran previamente incluidos en el grupo.

Cabe destacar que esta flexibilidad a la hora de cargar los alumnos puede permitir al administrador agrupar distintos grupos de matrícula sobre el mismo grupo de la aplicación o incluso partir el grupo original en varios subgrupos del sistema.

RF07

La aplicación debe permitir la posibilidad de dar de baja a un alumno de un grupo. Típicamente esto estará motivado por algún cambio de un alumno que implique dejar de estar matriculado en la asignatura.

El proceso de baja deberá ser abordado de dos formas:

- Individual: se selecciona un único alumno para su baja del grupo.
- Colectiva: se dan de baja todos los alumnos integrantes de un grupo.

Aparte de eliminar la pertenencia al grupo, este proceso debe eliminar el alumno del sistema en caso de que después de la baja del grupo no pertenezca a ningún grupo más presente en el sistema.

Este proceso debe controlar que el alumno no haya entregado/corregido ningún entregable, en cuyo caso no se permitirá la baja.

RF08

Se debe poder definir y gestionar prácticas. Para ello debemos definir una práctica como un conjunto de entregables que se identifica con una denominación establecida. Las operaciones permitidas para una práctica son:

- Alta de práctica: en este proceso se establece la descripción de la práctica.
- Baja de práctica: se selecciona la práctica a borrar del conjunto de prácticas definidas en el sistema.

Cabe señalar que el proceso de baja se llevará a cabo sí y solo sí:

- La práctica no tiene definidas entregas.
- La práctica no está asociada a ningún grupo.

En caso de que se incurra en una de estas dos circunstancias debe abortarse el proceso y notificarlo al usurario.

RF09

El sistema debe atender a la definición y borrado de entregas dentro de una práctica. Por ello debemos señalar que una entrega es una parte dentro de una práctica que viene determinada por:

- Descripción: una denominación que permita identificar su naturaleza de cara a los alumnos.
- Número de orden: el número correlativo que ocupa dentro del conjunto de entregas que componen la práctica.
- Fecha de inicio: fecha a partir de la cual se pueden entregar y corregir el conjunto de ejercicio o programas asociados a la entrega. La fecha estará compuesta de día, mes, año y hora con el fin de crear un intervalo de tiempo lo más definido posible.
- Fecha de fin: fecha hasta la cual los alumnos pueden entregar los ficheros asociados a la entrega. Al igual que la fecha de inicio, se compone de día, mes, año y hora.
- Método corrector a emplear: elemento que corregirá los ficheros subidos a la aplicación por los alumnos.

Respecto al borrado de una entrega se deberán presentar un listado de la totalidad de las prácticas definidas en el sistema con sus correspondientes entregables. A partir de este punto, se seleccionará la entrega a dar de baja. Este proceso debe llevarse a cabo sí y solo si no existen alumnos que hayan entregado/corregido la entrega.

RF10

El sistema debe permitir asociar las prácticas existentes en el sistema con los grupos definidos. De esto se deduce que podemos tener prácticas (con sus correspondientes entregas) que se encuentren disponibles para distintos grupos de alumnos.

RF11

Sobre la corrección de cada una de las entregas hay que decir que se debe hacer a través de elementos externos al propio sistema. Concretamente, cada una de las entregas definidas debe trabajar con servicio web cuyas operaciones disponibles sean referenciadas desde la aplicación. Esto nos debe permite tener un sistema versátil con distintos métodos de corrección, con un mínimo de cambios en la lógica de la aplicación.

El cómo estén implementados los servicios web no es objeto del presente proyecto, pero sí

debe serlo la interacción e integración del sistema con los mismos.

Aunque la corrección propiamente dicha no sea ejecutada por la aplicación, sí que se debe contemplar la opción de programar la hora en la que se lance un proceso que se encargue de interactuar con los distintos servicios web y obtener las correcciones de los ficheros entregados por los alumnos.

El administrador debe poder definir una hora en la que se ejecute a diario este proceso, que corregirá todo fichero enviado por los alumnos desde la última corrección ejecutada. Asimismo, el sistema enviará por correo electrónico una nota provisional de la entrega corregida. Esto constituye una fuente didáctica de gran valor, ya que el alumno puede ir comprobando sus progresos en la práctica a lo largo de todo el periodo de vigencia de la entrega.

RF12

Se establece que la aplicación genere una serie de informes para los distintos usuarios:

Administrador: accederá a un listado en el que se reflejen las notas obtenidas de todos los alumnos integrantes de un grupo para una entrega determinada. La práctica que contiene la entrega debe estar asociada al grupo. Los campos a mostrar serán los siguientes:

- Año académico
- Cuatrimestre
- Asignatura
- Grupo
- Nombre de grupo
- Nombre de la práctica
- Nombre de la entrega
- Rango de fechas de la entrega
- Datos de cada uno de los alumnos: nombre, apellidos, NIU, DNI y última nota de la entrega.

Alumno: obtiene un listado de las notas de una entrega determinada. El sistema debe presentar las entregas dentro de las prácticas asociadas a los grupos a los que pertenezca el alumno.

El contenido del listado será fundamentalmente el conjunto de notas que el alumno ha ido obteniendo con las sucesivas entregas de ficheros al sistema, por lo que puede ver la evolución que ha seguido durante el periodo de fechas en el que se define el entregable. Se deben contemplar los siguientes campos:

- Año académico
- Cuatrimestre
- Asignatura
- Nombre de la práctica
- Nombre de la entrega
- Rango de fechas de la entrega
- Datos del alumno: nombre, apellidos, NIU y DNI
- Datos de cada una de las entregas: nota y fecha de entrega.

RF13

La aplicación debe permitir al alumno subir ficheros al sistema asociándolos a una entrega determinada. Para ello, en primer lugar deben mostrarse el conjunto de los grupos del alumno junto con sus correspondientes prácticas y entregas asociadas para realizar la selección.

Una vez hecha la selección debe permitirse subir el conjunto de ficheros que constituyan la solución propuesta por el alumno. Se establece que este conjunto de ficheros sea subido en un único archivo comprimido en formato .zip. Una vez subido debe ser descomprimido y almacenado de forma automática en el sistema de ficheros.

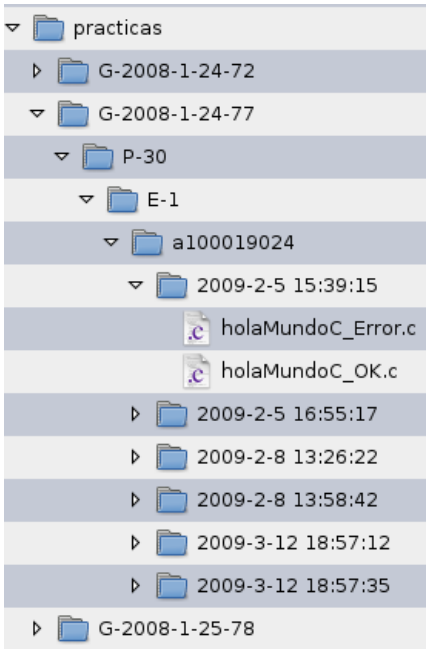
RF14

Debe contemplarse una estructura jerárquica de directorios que almacene el conjunto de ficheros subidos por los alumnos.

Dicha estructura debe organizarse en seis niveles:

- Nivel 1: grupo.
- Nivel 2: práctica.
- Nivel 3: entrega.
- Nivel 4: alumno
- Nivel 5: entrega de ficheros en una fecha determinada.
- Nivel 6: ficheros entregados.

En la siguiente figura se presenta un ejemplo de estructura válida de directorios:



De esta forma, vemos que el directorio raíz “practicas” contiene el conjunto de grupos definidos en el sistema. Uno de ellos sería el grupo “G-2008-1-24-77”. Éste tiene en su interior un directorio por cada una de las prácticas asociadas al grupo (en este caso sería “P-30”). Los directorios de las prácticas contienen uno por cada entrega integrante de las mismas (la práctica “P-30” contiene la entrega “E-1”). Dentro del directorio de la entrega encontramos el directorio del alumno (“a100019024”). Y ya dentro del directorio del alumno encontraríamos un directorio por cada vez que el alumno suba un conjunto de ficheros a la aplicación. Este último directorio sería el que contuviera los ejercicios.

3.3.2 Requisitos de interfaz de usuario

En este apartado se definen los requisitos relativos a la interfaz de usuario. De forma análoga a los requisitos funcionales se codifican con el acrónimo RI (Requisito de Interfaz) más un número correlativo.

RI01

El interfaz de la aplicación debe caracterizarse por ser claro, sencillo y eficaz. Asimismo, se recomienda que guarde una cierta armonía con el aspecto de las aplicaciones corporativas de la Universidad.

RI02

En todo momento, la aplicación debe contar con una cabecera situada en la parte superior de la pantalla. Esta cabecera deberá incluir necesariamente el logotipo del departamento.

RI03

En cualquier punto de la aplicación deben mostrarse los datos del usuario con el que se esté conectado a la aplicación. La situación será debajo de la cabecera de la aplicación.

Para el caso del alumno se mostrarán el nombre y los apellidos, mientras que para el administrador se mostrará el apelativo “ADMINISTRADOR”.

RI04

Las opciones de menú se mostrarán justo debajo de la identificación del usuario y se estructurarán de la siguiente manera:

- Barra de menú horizontal que permita acceder a los distintos menús específicos. Estará situada justo debajo de la identificación de usuario.

- Barras de menús específicos. Existirá una barra de menú con distintas opciones para datos personales, asignaturas, grupos, prácticas y listados de notas.

Los botones a utilizar deben ir en consonancia con el aspecto general de la aplicación.

RI05

Los mensajes de la aplicación dirigidos al usuario se deberán mostrar en la parte inferior de la pantalla y con la fuente de color rojo.

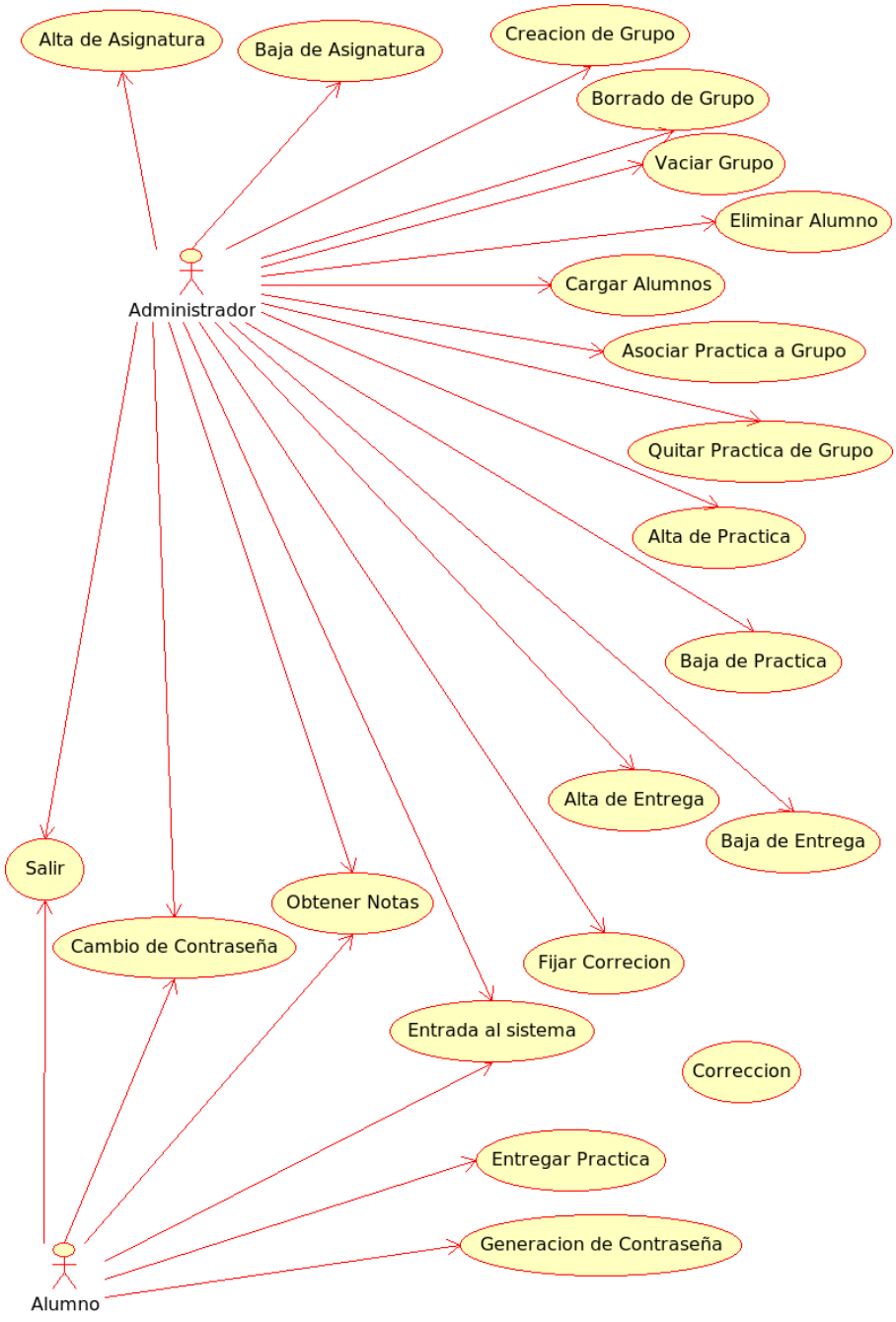
3.4 Modelado del sistema

A continuación se realiza el modelado del sistema. Este proceso consiste en describir el problema que se ha planteado utilizando unas reglas determinadas.

Para este proceso se ha recurrido al Lenguaje Unificado de Modelado (UML), que es un lenguaje gráfico que nos permite visualizar, especificar, construir y documentar un sistema software.

Para abordar de forma adecuada el modelado, en primer lugar se define el conjunto de casos de uso, que nos permite definir cómo interactúa el sistema con el usuario.

3.4.1 Diagrama de casos de uso



3.4.2 Documentación de los casos de uso

En este apartado se ofrece un análisis de cada uno de los casos de uso identificados. Para cada uno de ellos se detallan la descripción, los actores participantes, las precondiciones y postcondiciones, los flujos más comunes que seguiría el sistema y las excepciones que generan.

CU01	**Alta de Asignatura**
Actores:	Administrador
Descripción:	Se da de alta una asignatura en el sistema.
Precondiciones:	
Flujo principal:	• El administrador introduce la denominación de la asignatura. • Ejecución del alta.
Postcondiciones:	Existe una nueva asignatura en el sistema.
Excepciones:	

CU02	**Baja de Asignatura**
Actores:	Administrador
Descripción:	Se da de baja una asignatura del sistema.
Precondiciones:	Debe existir previamente una asignatura a dar de baja.
Flujo principal:	• Selección de la asignatura. • Ejecución de la baja.
Postcondiciones:	Se elimina la asignatura.
Excepciones:	E01 – La asignatura forma parte de un grupo previamente definido.

CU03	**Creación de Grupo**
Actores:	Administrador
Descripción:	Se da de alta un grupo.
Precondiciones:	Debe existir al menos un grupo creado.
Flujo principal:	• Selección de la asignatura. • Selección del año académico. • Selección del cuatrimestre. • Introducción del nombre del grupo. • Ejecución del alta.

Postcondiciones:	Se crea un nuevo grupo en el sistema.
Excepciones:	

CU04	**Borrado de Grupo**
Actores:	Administrador
Descripción:	Se da de baja un grupo.
Precondiciones:	Debe existir al menos un grupo creado.
Flujo principal:	• Selección del grupo • Ejecución de la baja
Postcondiciones:	El grupo se elimina del sistema.
Excepciones:	E02 - El grupo tiene prácticas asociadas. E03 - El grupo tiene alumnos cargados.

CU05	**Vaciar Grupo**
Actores:	Administrador
Descripción:	Se eliminan los alumnos de un grupo.
Precondiciones:	Debe existir al menos un grupo creado. El grupo debe terne alumnos cargados.
Flujo principal:	• Selección del grupo • Ejecución del vaciado.
Postcondiciones:	El grupo no tiene alumnos cargados.
Excepciones:	E04 - Los alumnos tienen entregadas/corregidas prácticas.

CU06	**Eliminar Alumno**
Actores:	Administrador
Descripción:	Se elimina un alumno del sistema.
Precondiciones:	El alumno debe haberse cargado previamente en algún grupo.
Flujo principal:	• Selección del grupo. • Selección del alumno. • Ejecución de la baja.
Postcondiciones:	El alumno ya no existe en el sistema.
Excepciones:	E05 - El alumno tiene entregada/corregida alguna práctica.

CU07	**Cargar Alumnos**
Actores:	**Administrador**
Descripción:	Se carga un conjunto de alumnos en un grupo.
Precondiciones:	Debe existir al menos un grupo creado.
Flujo principal:	• Selección del grupo. • Selección del fichero con los datos de los alumnos. • Ejecución de la carga.
Postcondiciones:	Los alumnos cargados se incorporan al grupo.
Excepciones:	E06 - Formato incorrecto del archivo a cargar.

CU08	**Alta de Práctica**
Actores:	Administrador
Descripción:	Se da de alta un práctica en el sistema.
Precondiciones:	
Flujo principal:	• Introducción de la denominación de las prácticas. • Ejecución del alta.
Postcondiciones:	La práctica queda almacenada en el sistema.
Excepciones:	

CU09	**Baja de Práctica**
Actores:	Administrador
Descripción:	Se elimina una práctica del sistema.
Precondiciones:	Debe existir al menos una práctica creada.
Flujo principal:	• Selección de la práctica. • Ejecución de la baja.
Postcondiciones:	La práctica queda eliminada.
Excepciones:	E07 - La práctica está asociada a un grupo E08 - La práctica tiene asociadas una o más entregas.

CU10	**Asociar Práctica a Grupo**
Actores:	Administrador
Descripción:	Se asocia una práctica a un agrupo para que los alumnos integrantes del mismo puedan corregir las distintas entregas de las que se compone la práctica.

Precondiciones:	Debe existir al menos una práctica. Debe existir al menos un grupo.
Flujo principal:	• Selección del grupo. • Selección de la práctica. • Ejecutar la asociación.
Postcondiciones:	Los alumnos del grupo en cuestión pueden corregir las entregas de las que se compone la práctica.
Excepciones:	

CU11	**Quitar Práctica de Grupo**
Actores:	Administrador
Descripción:	Se elimina la asociación entre un grupo y una práctica.
Precondiciones:	Existe una asociación previa entre un grupo y una práctica.
Flujo principal:	• Selección del grupo. • Selección de la práctica. • Ejecución de la baja.
Postcondiciones:	Se elimina la asociación.
Excepciones:	E09 - Existen alumnos del grupo que han corregido/entregado alguna entrega.

CU12	**Alta de Entrega**
Actores:	Administrador
Descripción:	Se de alta una entrega en una práctica determinada.
Precondiciones:	Existe al menos una práctica en el sistema.
Flujo principal:	• Selección de la práctica. • Introducción de la denominación de la entrega. • Introducción del orden de la entrega dentro de la práctica. • Introducción de la fecha de inicio de la entrega. • Introducción de la fecha de fin de la entrega. • Introducción de la url del servicio web que corrige la entrega. • Ejecución del alta.
Postcondiciones:	La práctica tiene una nueva entrega.
Excepciones:	E10 – Formato de los datos introducidos

CU13	**Baja de Entrega**
Actores:	Administrador

Descripción:	Se elimina una entrega de una práctica.
Precondiciones:	Existe al menos una entrega en la práctica.
Flujo principal:	• Selección de la práctica. • Selección de la entrega. • Ejecución de la baja.
Postcondiciones:	La práctica deja de tener disponible la entrega.
Excepciones:	E11 - Existen alumnos de cualquier grupo que han entregado/corregido la entrega.

CU14	**Fijar Corrección**
Actores:	Administrador
Descripción:	Se fija una hora para que se ejecute la corrección automática sobre todas las prácticas.
Precondiciones:	
Flujo principal:	• Introducción de la hora de corrección. • Actualización de la nueva hora de corrección.
Postcondiciones:	Existe una nueva hora en la que se corregirán las prácticas.
Excepciones:	E12 - Formato incorrecto de la hora.

CU15	**Entrada al Sistema**
Actores:	Administrador, Alumno
Descripción:	El actor en cuestión abre una sesión en el sistema introduciendo su identificador de usuario y su contraseña.
Precondiciones:	El actor está dado de alta en el sistema.
Flujo principal:	• Introducción del identificador de usuario. • Introducción de la contraseña. • Entrada al sistema.
Postcondiciones:	El actor ha entrado en el sistema y puede acceder a las opciones de menú.
Excepciones:	E13 - El actor introduce un valor incorrecto para el valor del usuario y/o la contraseña.

CU16	**Salir**
Actores:	Administrador, Alumno
Descripción:	El actor sale del sistema.

Precondiciones:	El actor ha abierto previamente una sesión en el sistema.
Flujo principal:	• Salida del sistema.
Postcondiciones:	
Excepciones:	

CU17	**Cambio de Contraseña**
Actores:	Administrador, Alumno
Descripción:	El actor cambia la contraseña con la que puede abrir una sesión en el sistema.
Precondiciones:	El actor tiene una sesión abierta en el sistema.
Flujo principal:	• Introducción de la contraseña antigua. • Introducción de la nueva contraseña. • Ejecución del cambio.
Postcondiciones:	El actor puede entrar al sistema con la nueva contraseña.
Excepciones:	E14 - Valores incorrectos de la contraseña antigua y/o nueva.

CU18	**Obtener Notas**
Actores:	Administrador, Alumno
Descripción:	El actor obtiene un listado de notas en función de los parámetros introducidos por pantalla. Según el tipo de actor obtendrá: • Administrador: un listado de las notas de todos los alumnos de un grupo que han ejecutado una entrega. • Alumno: listado de las notas de una entrega determinada
Precondiciones:	Existen entregas corregidas.
Flujo principal:	• Selección del grupo. • Selección de la práctica. • Selección de la entrega.
Postcondiciones:	Se obtiene el listado en formato pdf.
Excepciones:	

CU19	**Entrega de Práctica**
Actores:	Alumno
Descripción:	El alumno sobe un conjunto de ficheros asociados a una entrega para que sean corregidos posteriormente.
Precondiciones:	El alumno ha abierto una sesión en el sistema. El alumno está cargado en al menos un grupo.

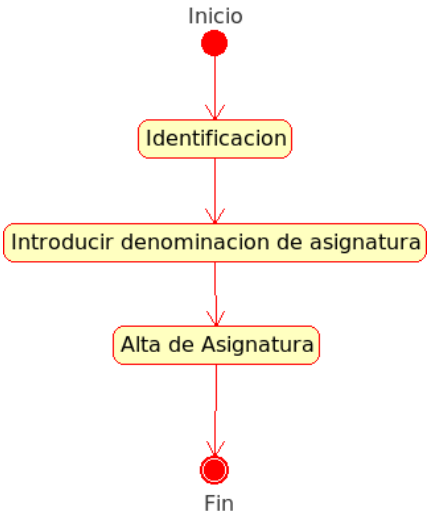
	El grupo al que pertenece el alumno debe tener asociada al menos una práctica. La práctica debe estar compuesta por al menos una entrega.
Flujo principal:	• Selección del grupo. • Selección de la práctica. • Selección de la entrega. • Introducción del fichero comprimido con los archivos a corregir. • Ejecutar la entrega de ficheros.
Postcondiciones:	El conjunto de ficheros subido por el alumno se almacena en el sistema de ficheros de la aplicación.
Excepciones:	E15 - Formato incorrecto del archivo comprimido que contiene los ficheros.

CU20	**Generación de Contraseña**
Actores:	Alumno
Descripción:	El alumno pide al sistema que genera una nueva contraseña para poder entrar establecer una sesión válida. La aplicación la envía al correo electrónico que el alumno ha introducido después de haberla generado.
Precondiciones:	El alumno ha sido cargado a un grupo válido.
Flujo principal:	Introducción del correo electrónico. Ejecución de la generación de contraseña.
Postcondiciones:	El alumno cuenta con una nueva contraseña.
Excepciones:	E16 - Formato incorrecto del correo electrónico.

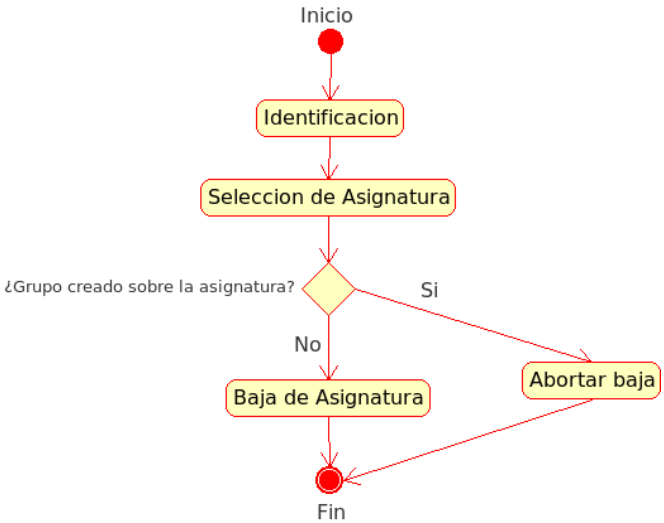
3.4.3 Diagramas de actividad

En este apartado se presentan los diagramas de actividad para cada uno de los casos de uso identificados.

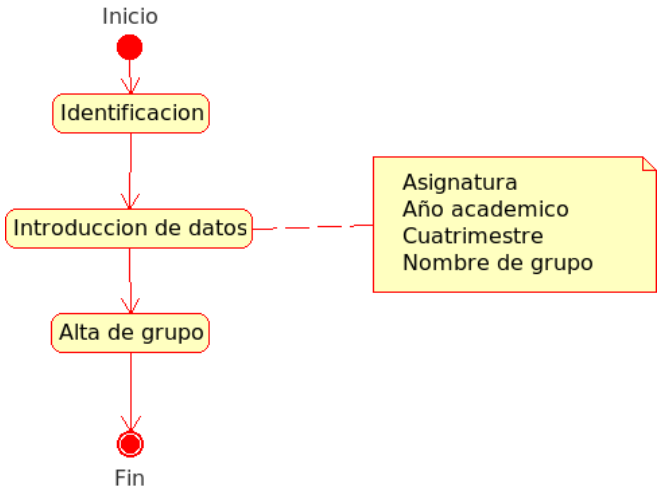
CU01 Alta de Asignatura



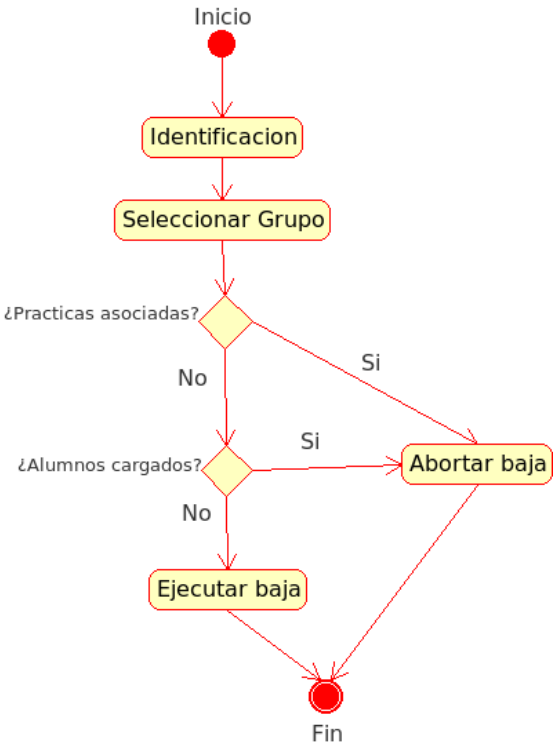
CU02 Baja de asignatura



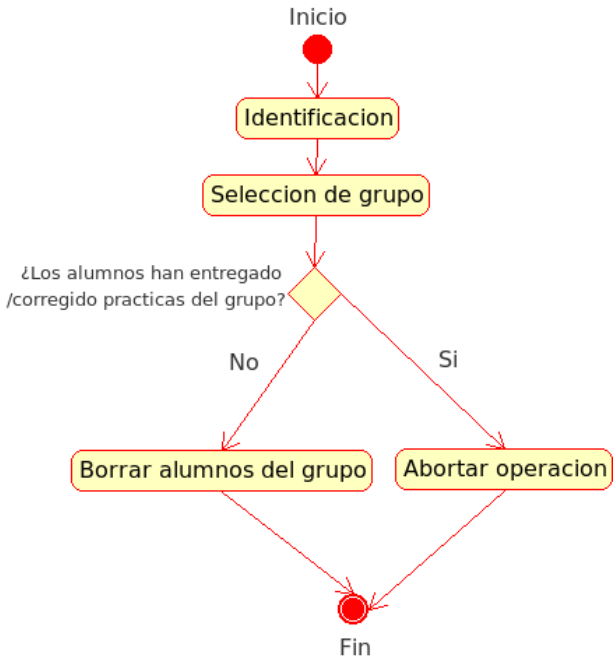
CU03 Creación de grupo



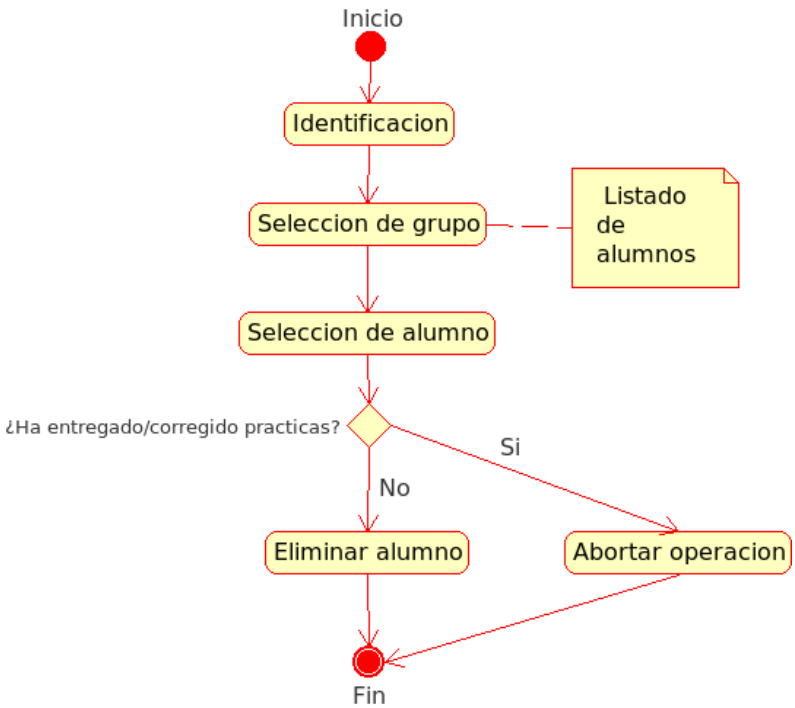
CU04 Borrado de grupo



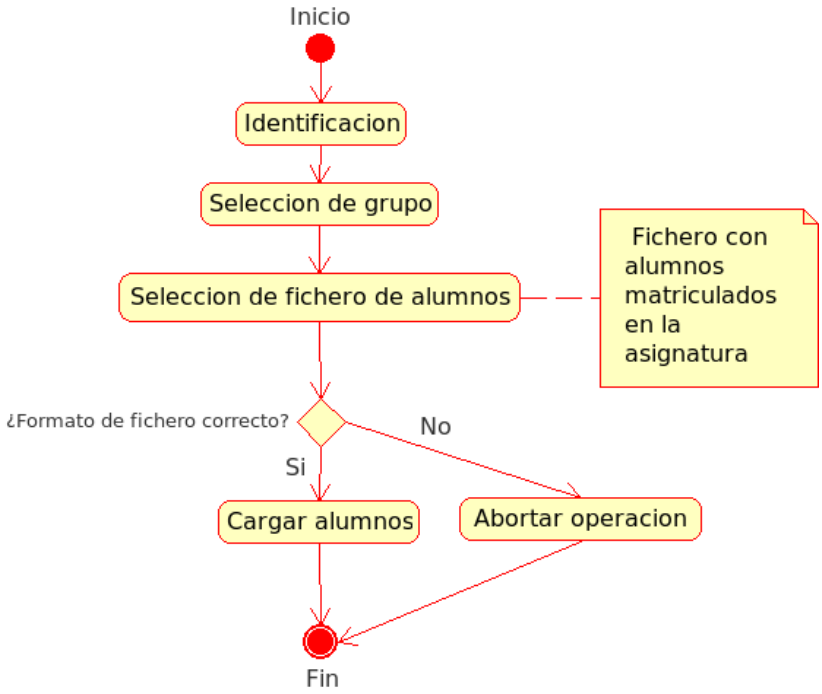
CU05 Vaciar Grupo



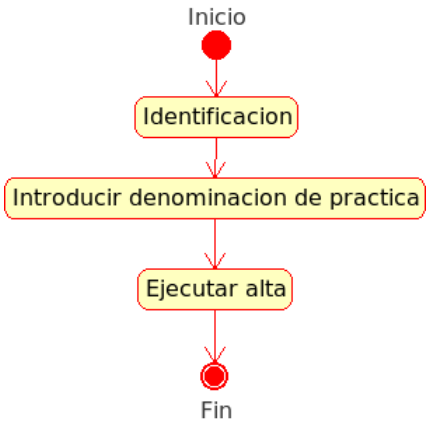
CU06 Eliminar Alumno



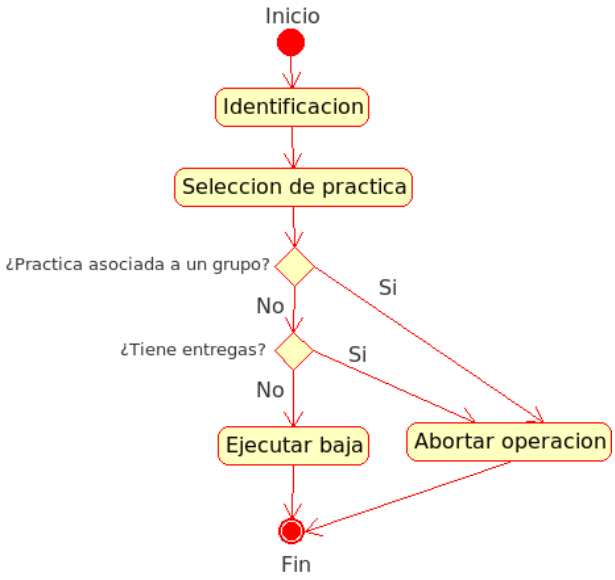
CU07 Cargar Alumnos



CU08 Alta de Práctica

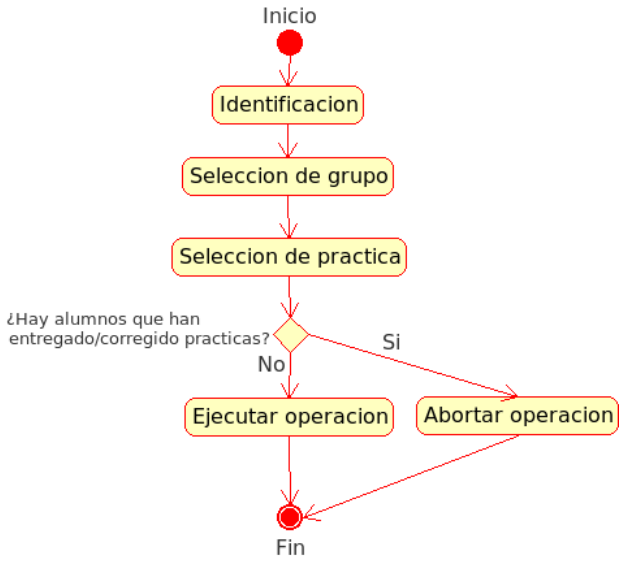


CU09 Baja de Práctica

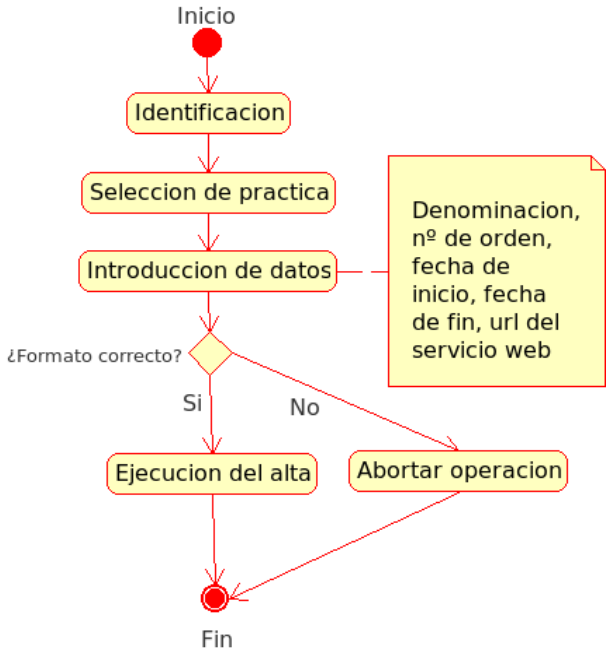


CU10 Asociar Práctica a Grupo

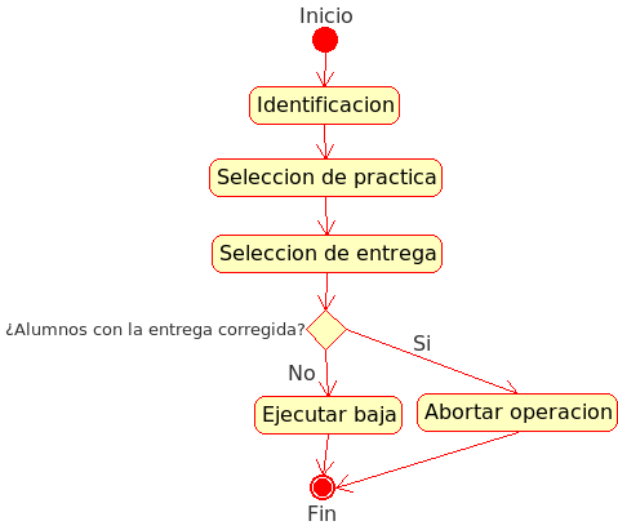
CU11 Quitar Práctica de Grupo



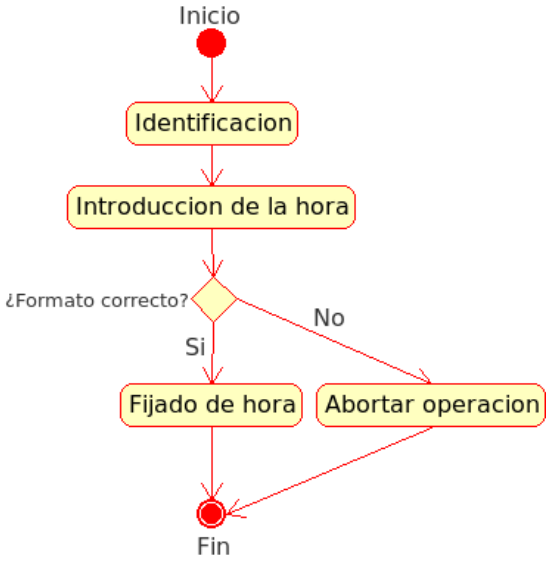
CU12 Alta de Entrega



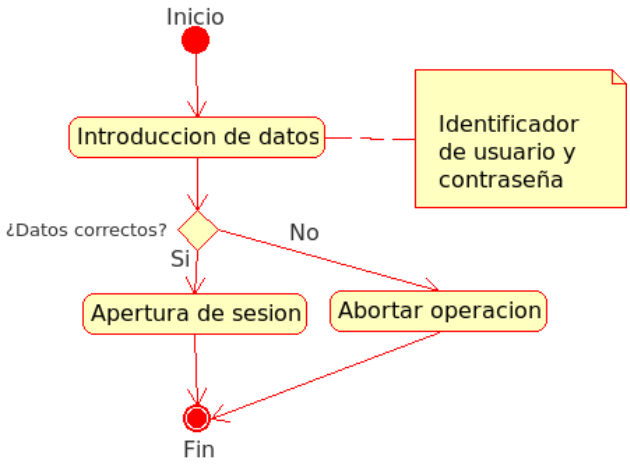
CU13 Baja de Entrega



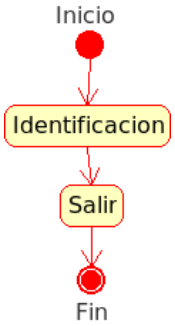
CU14 Fijar Corrección



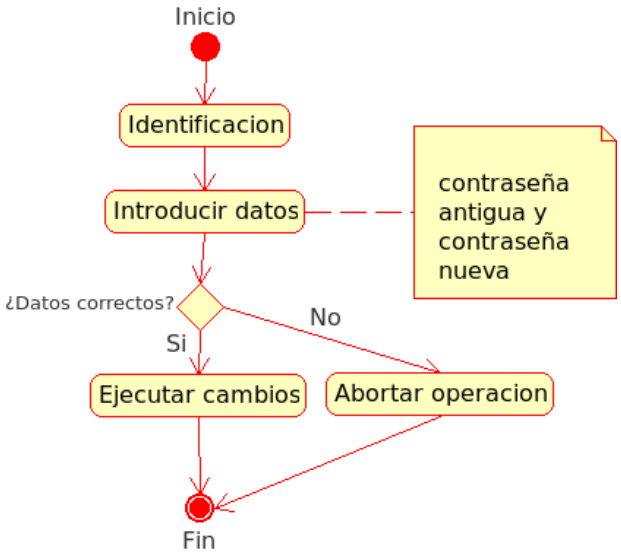
CU15 Entrada al Sistema



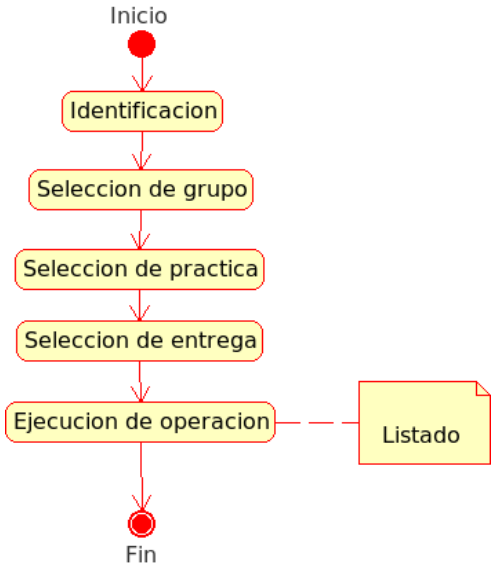
CU16 Salir



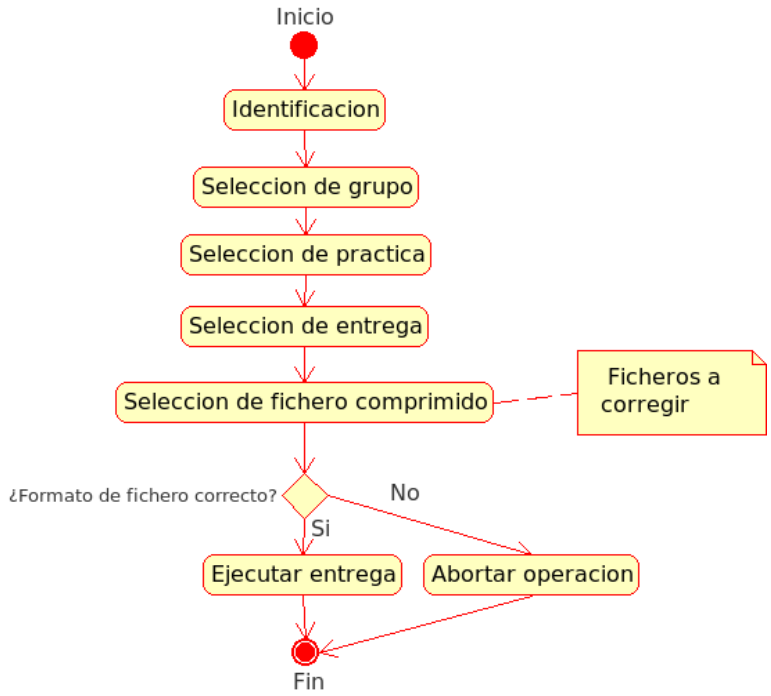
CU17 Cambio de Contraseña



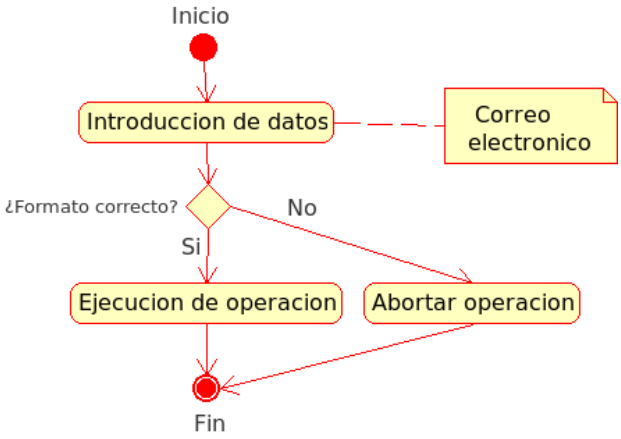
CU18 Obtener notas



CU19 Entrega de Práctica

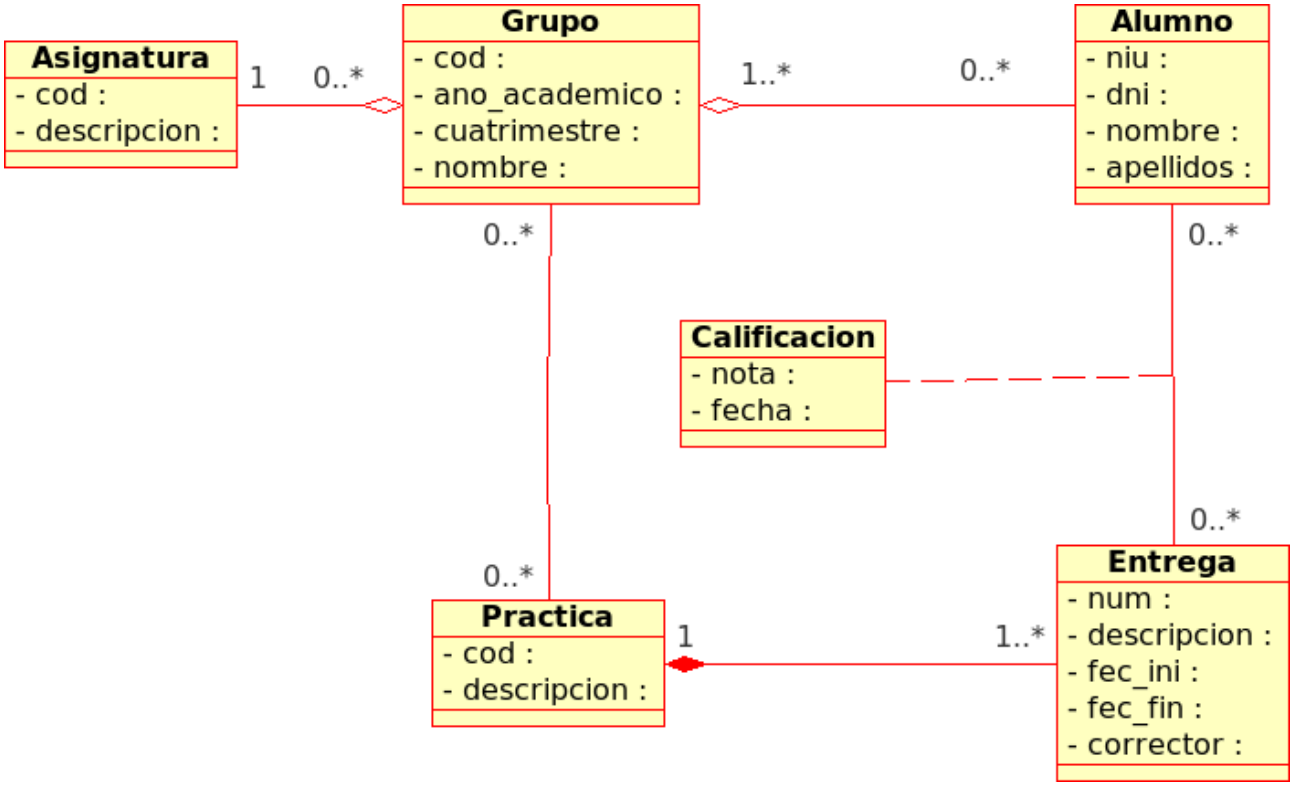


CU20 Generación de contraseña



3.4.4 Diagrama de clases

A continuación se presenta el diagrama de clases, que tiene la finalidad de representar el dominio del problema. Asimismo se comentará por separado cada una de las clases identificadas y las distintas asociaciones presentes en el diagrama.



CLASE *Asignatura*

Esta clase representa las asignaturas en las que están matriculados los alumnos. Se definen los siguientes atributos:

- cod: código que identifica unívocamente a la asignatura.
- descripción: denominación de la asignatura

CLASE *Grupo*

Entidad que representa un conjunto de alumnos matriculado en una asignatura. Tiene los siguientes atributos:

- cod: código que identifica el grupo.
- ano_academico: año académico para el que se define el grupo.
- cuatrimestre: cuatrimestre en el que se ubica el grupo.
- nombre: denominación impuesta al grupo

CLASE *Alumno*

Esta clase representa a los alumnos matriculados en las distintas asignaturas y que van a realizar el conjunto de prácticas propuestas. Cuenta con los siguientes atributos:

- niu: Número de Identificación de Usuario. Es el número asignado por la Universidad en el momento de matricularse por primera vez. Es el usuario con el que el alumno va a conectarse al sistema.
- dni: Documento Nacional de Identidad del alumno o número de pasaporte para alumnos extranjeros.
- nombre: nombre del alumno.
- apellidos: apellidos del alumno.

CLASE *Práctica*

Representa unas prácticas docentes definidas que van a estar compuestas de un conjunto de entregables. Cuenta con los siguientes atributos:

- cod: identificador unívoco de la práctica.
- descripción: denominación formal de las prácticas.

CLASE *Entrega*

Esta clase representa cada uno de los entregables de los que se compone una práctica. Su existencia no tiene sentido por sí misma, ya que siempre está asociada a una práctica en concreto. Se definen los siguientes atributos:

- num: número de entrega dentro del conjunto de entregas de una práctica.
- descripción: denominación formal de la entrega.
- fec_ini: fecha a partir de la cual los alumnos pueden entregar la práctica.
- fec_fin: fecha hasta la cual los alumnos pueden entregar la práctica.
- corrector: url del servicio web que se va a encargar de corregir la entrega.

CLASE *Calificación*

Representa cada una de las calificaciones de las distintas entregas realizadas por los alumnos. Contiene los siguientes atributos:

- nota: valoración del entregable hecho por el alumno según el corrector utilizado.
- fecha: fecha en la que el alumno ha realizado la entrega.

ASOCIACIÓN entre ***Grupo*** y ***Asignatura***

Esta asociación determina una de las características de un grupo, que es su relación con una única asignatura. Sin embargo, hay que decir que una misma asignatura puede pertenecer a distintos grupos de matrícula, hecho característico del ámbito académico.

ASOCIACIÓN entre ***Grupo*** y ***Alumno***

Esta asociación representa el conjunto de alumnos que van a formar parte de un grupo determinado. Un mismo alumno puede ser parte integrante de varios grupos, ya que cada grupo está asociado a una única asignatura.

ASOCIACIÓN entre ***Grupo*** y ***Practica***

Representa el conjunto de prácticas que están disponibles para un grupo determinado. Así, todos los alumnos integrantes de un grupo podrán acceder al conjunto de prácticas relacionadas con el mismo.

ASOCIACIÓN entre ***Practica*** y ***Entrega***

Representa el conjunto de entregables que forman parte de una práctica. Una entrega o entregable no determina su existencia por sí misma, sino que siempre debe estar asociada a una práctica concreta. De ahí que se establezca una relación de composición entre las dos clases.

ASOCIACIÓN entre ***Alumno*** y ***Entrega***

Esta asociación representa las entregas de ficheros que realizan los alumnos asociadas a una entrega determinada. La relación lleva asociada la clase Calificación, ya que para cada una de las entregas realizadas en una fecha el alumno obtiene una nota.

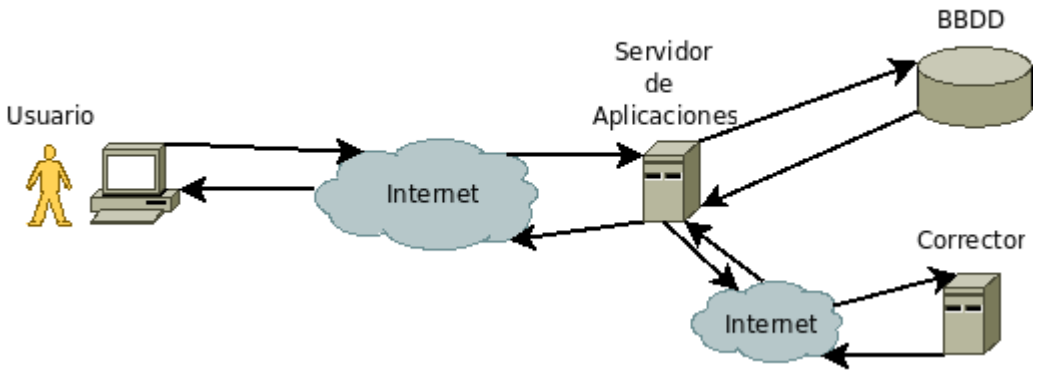
CAPÍTULO 4:

DISEÑO

4 Diseño

4.1 Arquitectura preliminar

En este apartado se presenta la arquitectura preliminar a implementar. La arquitectura definitiva se definirá después de hacer un estudio tecnológico sobre las distintas partes del esquema. A continuación se presenta un gráfico con todos los elementos integrantes del sistema.



En el esquema identificamos los siguientes elementos:

- Usuario: representa al administrador de la aplicación y al alumno, que van a interactuar con el sistema de la misma forma, a través de un navegador web. Con él accederán a la aplicación alojada en el servidor web y ejecutarán un conjunto de operaciones determinadas.
- Servidor de aplicaciones: elemento que almacenará la aplicación software que va a implementar las funcionalidades requeridas.
- Base de datos: su función en esta arquitectura es la de almacenar los datos que requiera la aplicación.
- Corrector: elemento externo a la aplicación que se encarga de corregir los ficheros que le son enviados desde la aplicación.

4.2 Estudio tecnológico y selección de tecnologías

Una vez definidos los elementos básicos del sistema, procede a realizarse un estudio de cada uno de ellos considerando distintas alternativas tecnológicas que se pueden utilizar. Para cada una de ellas se evaluarán sus pros y sus contras, características que determinarán la elección final.

4.2.1 Navegador web

La elección de una navegador web determinado no se considera relevante en este contexto, ya que el imponer un determinado navegador para acceder una aplicación web viola la capacidad de elección del usuario y no haría de la aplicación un elemento que respetase los estándares, algo que se considera fundamental en el entorno web.

De esta forma se considera que la aplicación debe ser accesible desde los navegadores web más conocidos del mercado: Internet Explorer, Mozilla Firefox, Opera y Safari.

4.2.2 Tecnologías a ejecutar en el servidor

En este punto se consideran distintas tecnologías a utilizar en la parte del servidor de aplicaciones. Este punto es extremadamente importante, ya que su elección va a determinar el tipo de servidor de aplicaciones y el sistema gestor de bases de datos a emplear.

Inicialmente se proponen las siguientes tecnologías: CGI, ASP, ASP.NET, PHP, Java Servlets y JSP.

CGI (Common Gateway Interface)

Interfaz Común de Pasarela. Los CGI son programas que se ejecutan en la parte del servidor. Pueden ser implementados en distintos lenguajes, entre los que destaca Perl. Tienen el principal inconveniente que consumen muchos recursos del servidor al generarse una instancia del CGI para cada usuario que lo ejecuta, hecho que puede comprometer el uso de memoria del servidor.

Si se desarrollan en un lenguaje que requiera compilación antes de su ejecución pueden presentar problemas de portabilidad al depender de un compilador determinado. Por ello una opción razonable para su desarrollo es el lenguaje interpretado Perl.

Una ventaja puede ser la facilidad de utilizar múltiples lenguajes para su desarrollo, pero el consumo excesivo de recursos hace que se descarte su uso en el presente proyecto.

ASP (Active Server Pages)

Páginas Activas del Servidor. Es un lenguaje desarrollado por Microsoft que posee la característica de ser compatible con todos los navegadores web, ya que al ser interpretado la página resultante de la solicitud de un navegador será HTML. Concretamente se trata de rutinas a modo de sintaxis de script que son llamadas desde la propia página HTML, hecho que servirá para completar el contenido dinámico de la página.

De forma nativa sólo es compatible con el servidor Web IIS, por lo que se requiere de módulos externos para funcionar en otros servidores. Ésto y el hecho que desde 2002 esté siendo sustituido por ASP.NET hace que igualmente se descarte su uso.

ASP.NET

Se trata de la evolución de ASP para la plataforma .NET de Microsoft. A grandes rasgos, el cambio fundamental que hace respecto al ASP clásico es implantar el lado del cliente en el lado del servidor, hecho que permite a los desarrolladores un mayor control sobre los documentos dinámicos. Utiliza la plataforma .NET, que incluye objetos de formulario a medida a través de los cuales el cliente interactúa con el servido.

Se trata de un entorno orientando a objetos bastante potente que contempla una amplia variedad de lenguajes y que permite una gran selección de componentes y recursos ofrecidos por la plataforma .NET.

Como desventajas cabe señalar que uso uso está limitado a las plataformas ofrecidas por Microsoft, hecho que determina la baja portabilidad de una aplicación desarrollada en esta tecnología. Respecto al uso cuenta con una licencia privativa, hecho que determina la opacidad de la plataforma. Finalmente se ha optado por descartarla como tecnología a emplear.

PHP (PHP Hypertext Pre-processor)

PHP es un lenguaje de programación interpretado, diseñado para ser ejecutado en el lado del servidor y orientado inicialmente para la creación de páginas web dinámicas. Su licencia permite el libre uso y distribución y es multiplataforma.

Es capaz de interoperar con la mayoría de sistema gestores de bases de datos. Además, su curva de aprendizaje es bastante baja.

Sin embargo, PHP es un lenguaje débilmente tipado y la orientación a objetos, si bien ha hecho grandes avances desde la versión 5, tiene todavía mucho camino que recorrer. Por ello también debemos descartar su uso.

Java Servlets

Los servlets son objetos que se ejecutan en un servidor o contenedor JEE (Java Enterprise Edition) y están diseñados para ofrecer contenido dinámico desde un servidor web, generalmente HTML. Pueden ser ejecutados en cualquier servidor debido a la tecnología Java que los implementa. Los Servlets incrementan la funcionalidad de una aplicación web. Se cargan de forma dinámica por el entorno de ejecución Java del servidor cuando se necesitan. Cuando se recibe una petición del cliente, el contenedor/servidor web inicia el servlet requerido. El Servlet procesa la petición del cliente y envía la respuesta de vuelta al contenedor/servidor, que es enviada al cliente.

En el protocolo HTTP existen dos tipos de mensajes: request y response, que se suceden síncronamente (no hay un response sin un request previo y no puede haber más de un request consecutivo sin su response previo y viceversa). Los servlets Java utilizan estos mensajes a través de los interfaces HttpServletRequest y HttpServletResponse, que encapsulan respectivamente los mensajes request y response.

La API de programación de los servlets hace muy fácil la escritura de servicios complejos para aplicaciones basadas en web, sin tener que centrarse en los detalles de bajo nivel de los protocolos HTTP, formatos de petición, y cabeceras. Su uso en aplicaciones web implica una mejora sustancial en la gestión de recursos, ya que no se genera un único proceso en el servidor

cada vez que llega una petición del cliente, sino que se crea un hilo ligero. Este hilo ligero es un proceso hijo controlado por por el proceso padre (el servidor). En este escenario, el cambio de contexto entre procesos se hace muy fácil, y los hilos pueden pasar fácilmente de activos a inactivos, o a espera. Esto mejora sustancialmente el rendimiento de los servlets sobre los scripts CGI.

Este tipo de funcionamiento, su independencia de la plataforma utilizada y el alto soporte de excepciones hace que se considere esta tecnología para implementar la aplicación.

JSP (Java Server Pages)

JSP es una tecnología de Sun Microsystems que permite generar páginas web dinámicas. A grandes rasgos consiste en páginas HTML con código Java incrustado. Esto hace que antes de su ejecución requieran ser compiladas en el servidor que las contenga.

Pueden ser vistas como una abstracción de alto nivel de los servlets, ya que después de compilada, una jsp es convertida en un servlet, con todas las ventajas que implica, por lo que también se considera su uso para la implementación de la aplicación.

Las ventajas que nos ofrecen tanto los servlets Java, como las páginas JSP, llevan a concluir el uso de la tecnología Java EE (Enterprise Edition) en el presente proyecto, decisión motivada tanto por el alto rendimiento de la tecnología en entornos web como de abundancia de documentación y recursos.

4.2.3 Servidor de aplicaciones

Dado que se ha tomado la decisión de utilizar la tecnología Java EE (Enterprise Edition) se ha optado por utilizar Apache Tomcat como elemento contenedor de la aplicación a implementar.

Tomcat es un contenedor de servlets desarrollado por la Apache Software Foundation. Implementa tanto las especificaciones de las JSP (Java Server Pages) como las de los servlets Java. Proporciona un entorno web alojado en un servidor que permite ejecutar código java.

La aplicación incluye distintas herramientas para gestionar y configurar el entorno, pero también lo permite editando sus ficheros de configuración en xml.

Sus características más importantes son:

- Multiplataforma: puede funcionar en distintos entornos como Linux, Solaris, Windows...
- Soporte: tiene a una entidad de renombre soportando su desarrollo, la Apache Software Foundation, una garantía de calidad y estabilidad.
- Licencia: se licencia como código abierto, concretamente con la licencia Apache, la cual permite el uso y distribución del código fuente tanto para software libre como para software propietario. La única condición es que se mantenga el aviso de copyright y el disclaimer.
- Estabilidad: tomcat lleva desarrollándose desde hace bastante tiempo (la primera versión data de 2001).

Tomcat tiene una estructura modular en la que se distinguen los siguientes elementos:

- Catalina, el contenedor de servlets.
- Coyote, la parte que soporta el protocola HTTP 1.1
- Jasper 2, el módulo que se encarga de procesar las páginas JSP.

Sin embargo en Tomcat no son todo ventajas. Por ejemplo, no soportan Enterprise Java Beans (EJB) como otras soluciones de código abierto como Jboss o Glassfish. Sin embargo, la elección de Tomcat como contenedor de la aplicación está motivada en que soporta perfectamente las tecnologías que se van a emplear en el presente proyecto y que consume menos recursos que otros servidores de aplicaciones más versátiles.

4.2.4 Base de datos

En este apartado se realiza una comparativa entre los sistemas gestores de bases de datos más comunes. Para ello se comentarán de forma breve y posteriormente se presentará una tabla con sus características fundamentales enfrentadas.

Las bases de datos deben presentar compatibilidad con el lenguaje de programación seleccionado en el punto anterior. De igual forma, constituirán puntos a favor tales como el soporte multiplataforma, la escalabilidad, la estabilidad, el soporte transaccional, el tipo de licencia y la seguridad del sistema.

DB2

Es un motor de base de datos relacional, propiedad de IBM y con licencia privativa, que integra XML de manera nativa, es multiplataforma y con soporte empresarial. El principal defecto que tiene es que no se pueden hacer consultas SQL muy grandes. Tiene un límite muy pequeño para asignar nombre de variables tanto en tablas como en columnas.

Oracle

Motor de base de datos relacional, propiedad de Oracle, con licencia privativa, multiplataforma y con soporte empresarial. Tiene una gran soporte transaccional, gran robustez, escalabilidad y es altamente usado en el mundo empresarial.

SQL Server

Sistema gestor de base de datos desarrollado por Microsoft. Su uso está limitado a un único usuario. No es multiplataforma, ya que solo está disponible para los sistema operativos de Microsoft, pero es estable, escalable y tiene gran soporte transaccional.

Access

Es un sistema gestor de base de datos relacional desarrollado por Microsoft con licencia privativa. Es altamente usado en empresas pequeñas para proyectos de pequeña envergadura. Solo está disponible para los sistemas operativos de Microsoft. No soporta transacciones y es poco escalable.

MySQL

Es un sistema gestor de base de datos relacional desarrollado originalmente por MySQL AB,

perteneciendo actualmente a Sun Microsystems. Posee doble licencia, una GPL y otra para uso comercial. Es multiplataforma, soporta gran contenido de transacciones y es muy usado a nivel empresarial. Es perfectamente escalable y estable.

SGBDR	Soporte multilenguaje	Soporte multiplataforma	Escalabilidad	Estabilidad	Licencia GPL	Soporte transaccional
DB2	Sí	Sí	Alta	Alta	No	Alto
Oracle	Sí	Sí	Alta	Alta	No	Alto
SQL Server	Sí	No	Alta	Alta	No	Alto
Access	Sí	No	Baja	Alta	No	No
MySQL	Sí	Sí	Alta	Alta	Sí	Alto

Después de esta comparativa, se ha tomado la decisión de utilizar MySQL como el sistema gestor que aloje la base de datos de la aplicación. Esto es debido a que reúne las características esenciales que deben exigirle a un sistema gestor de base de datos: soporte multilenguaje, soporte multiplataforma, escalabilidad, estabilidad y soporte transaccional. Adicionalmente consideramos que el tipo de licencia con el que se distribuye es otro punto más a su favor.

4.2.5 Corrector

En este punto se determinará la tecnología a utilizar para el elemento corrector del sistema. En primer lugar hay que decir que se nos impuso el uso de la tecnología de los servicios web para este punto, por lo que a continuación se describirá de forma concisa en qué consiste un servicio web.

Los servicios web son tecnologías que comprenden un conjunto de protocolos y estándares para intercambiar datos entre distintas aplicaciones. Lo interesante de este intercambio es que puede realizarse entre aplicaciones desarrolladas en lenguajes de programación distintos. Así, una aplicación que interaccione con un servicio web puede ejecutar los procedimientos ofrecidos por éste y no se tiene que preocupar en su implementación, únicamente tiene que ajustarse a las normas que rigen los servicios web.

Los servicios web comprenden los siguientes protocolos:

- ***XML*** (Extensible Markup Language): Es el formato en el que se intercambian los datos.
- ***SOAP*** (Simple Object Access Protocol): protocolos sobre los que se establece el intercambio.
- ***WSDL*** (Web Services Description Language): lenguaje utilizado para describir los servicios.
- ***UDDI*** (Universal Description, Discovery and Integration): sistema para publicar y comprobar la disponibilidad de los servicios web.

Algunas ventajas:

- Alta flexibilidad debido a la transparencia de la plataforma en la que está implementado el servicio web, hecho solo posible debido al uso de protocolos estándares.
- Descentralización y distribución, características que hacen posible que sean accedidos desde distintos puntos de la red.
- Fácil integración y reutilización en distintos proyectos.

Estos argumentos hacen que su integración en este proyecto constituya una oportunidad idónea para comprobar su alto grado de integración e interoperabilidad.

Para el presente proyecto se ha tomado la decisión de emplear Apache Axis2 como elemento que albergue los servicios web, ya que su integración con el servidor de aplicaciones seleccionado, Apache Tomcat, es muy sencilla. La definición de los servicios web sobre Axis2 se hace definiendo clases java sencillas en integrándolas en la estructura de directorios de Axis2. Estas clases no dependen de ningún framework y se conocen comúnmente por POJO (Plain Old Java Object).

4.3 Arquitectura definitiva

A continuación se presenta un diagrama con la arquitectura definitiva del sistema.

Hay que señalar que en servidor de aplicaciones Apache Tomcat vamos a tener desplegados dos elementos:

- Aplicación a implementar: para su codificación se emplea el patrón Modelo-Vista-Controlador (MVC), elemento que se detallará en el siguiente apartado.
- Axis 2: Contiene el elemento corrector.

4.4 Patrón MVC

El patrón Modelo-Vista-Controlador es un patrón de arquitectura software. Se caracteriza porque separa una aplicación en tres componentes principales:

- El controlador, único elemento que puede recibir acciones de los usuarios externos.
- El modelo, que posee el estado interno de la aplicación (determinado por el estado de sus entidades) y las reglas de negocio.
- Las vistas, que reflejan el estado del modelo en un momento dado.

Este patrón constituye la forma lógica en la que deben diseñarse e implementarse las aplicaciones web. Su uso tiene múltiples ventajas:

- Clara separación entre interfaz, lógica de negocio y de presentación, lo que permite un mejor desarrollo del software.
- Sencillez para crear distintas representaciones de los mismos datos.
- Facilidad para la realización de pruebas unitarias de los componentes.
- Reutilización de los componentes.
- Simplicidad en el mantenimiento de los sistemas.
- Facilidad para desarrollar prototipos rápidos.

- Los desarrollos suelen ser más escalables.

4.4.1 Struts

Actualmente, Struts es uno de los frameworks más utilizados a la hora de trabajar con el patrón MVC. Se distribuye bajo la licencia Apache, por lo que es *Open Source.*

El funcionamiento de las revisiones de la versión 2.0 del framework difieren sustancialmente de las revisiones de la versión 1.0.

Struts1

Struts1 se basa en la existencia de un servlet (Front Controller) que actúa de controlador central y recibe todas las peticiones generadas por el usuario. A partir de la dirección de origen, de los datos de entrada y del estado actual de la aplicación, selecciona la vista apropiada a mostrar.

Tiene las siguientes características principales:

- Lógica de navegación entre páginas.
- Binding entre java y el código html.
- Validación de entradas.
- Internacionalización.
- Independencia del motor de visualización.
- Maquetación.

El funcionamiento básico seguiría la siguiente secuencia:

1. El cliente solicita una página que contiene datos a completar.
2. El servidor le envía la página.
3. El cliente, con los datos completados envía de regreso la página. El controlador verifica la ruta con la que se invocó, extrae el path de esa ruta y busca en el conjunto de mapeos de la aplicación cual es la acción a invocar y qué formulario necesita recibir como entrada.

4. El controlador crea o reutiliza el formulario dependiendo del ámbito el que se ejecute la petición, carga los datos en el formulario, los valida, crea la acción y pasa el formulario como parámetro.
5. La acción recibe el formulario y con sus datos invoca las reglas de negocio (generalmente delegadas en otros objetos).
6. A partir de la respuesta recibida, se cargan los valores de salida y se selecciona la siguiente vista a mostrar.

Struts2

En primer lugar hay que señalar que la versión 2 de Struts está basada en el ya existente framework WebWork2, y el producto final nace de la colaboración entre las comunidades de ambos proyectos. El objetivo de Struts 2 es el mismo que el de Struts 1, ofrecer soporte para el desarrollo de aplicaciones web bajo el patrón MVC.

Aunque Struts 2 es un desarrollo nuevo, posee algunas similitudes con la versión 1, ya que las dos proporcionan tres componentes:

- Un manejador de peticiones (request handler) que mapea clases Java a URIs de aplicaciones web.
- Un manejador de respuestas (response handler) que mapea nombres lógicos a páginas existentes en el servidor u otros recursos web.
- Un conjunto de etiquetas que permiten crear aplicaciones robustas basadas en formularios.

A continuación se detallan los cambios fundamentales de Struts 2 frente a Struts 1:

- Mejoras del diseño: Todas las clases de Struts 2 están basadas en interfaces. Las interfaces del núcleo son independientes de la arquitectura HTTP.
- Resultados mejorados: A diferencia de los antiguos “ActionForward” de Struts 1, los resultados de Struts 2 son tipificados permitiendo preparar el response en los casos que sea necesario.

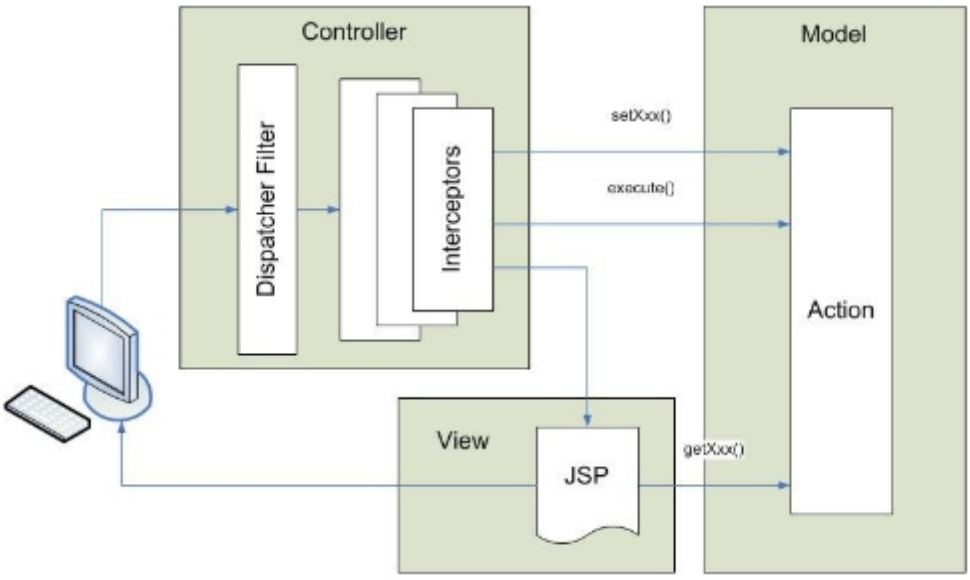
- Mejoras en las etiquetas: Con las nuevas etiquetas de Struts 2 es posible crear páginas consistentes con menor escritura de código.
- Soporte para Ajax: El "theme" ajax permite a las aplicaciones interactuar con el servidor mediante requerimientos asíncronos.
- Facilidad de cambio: Muchos cambios pueden ser realizados dinámicamente sin ser necesario el reinicio del contenedor web.
- Pruebas de las acciones: Las acciones de Struts 2 son independiente de la arquitectura HTTP con lo cual pueden ser fácilmente probadas mediante test de unidad sin la necesidad de recurrir a objetos que simulen el comportamiento de los objetos reales.
- Plugins: Las extensiones de Struts 2 pueden ser agregadas simplemente copiando un fichero con extensión ".jar". No es necesaria una configuración manual.
- Formularios: En Struts 2 no existe más el concepto de formulario tal como existía en Struts 1. Ahora es posible usar cualquier JavaBeans contenido en una acción o escribir directamente sobre los atributos de una acción. Por otro lado, Struts 2 convierte automáticamente los valores cargados en el formulario web a los tipos de valores que posea el JavaBean referenciado.
- Acciones POJO (Plain Old Java Object): Cualquier clase puede ser usada como una acción, por lo que ya no es necesario implementar una interfaz.

En la siguiente tabla se comparan las características más importantes de las dos versiones de Struts:

Característica	***Struts 1 (S1)***	***Struts 2 (S2)***
Acciones	S1 requiere que las clases Action extiendan una clase abstracta propia del framework. Un problema común en S1 es programar con clases abstractas en vez de interfaces	Un Action en S2 puede implementar una interface Action, entre otras interfaces para habilitar servicios opcionales. S2 provee una clase base "ActionSupport" la cual implementa las interfaces más comunes.
Modelo de threads	Las acciones en S1 son singletons y deben ser "thread-safe" dado que hay una única instancia de la clase que maneja todos los requerimientos. La estrategia del uso de singletons agrega restricciones sobre las cosas que se pueden hacer con S1. Los recursos de una acción deben ser "thread-safe" o "synchronized".	Los objetos de las acciones de S2 son instanciados por cada "request", con lo cual no existen problemas de "thread-safety".

Característica	*Struts 1 (S1)*	*Struts 2 (S2)*
Dependencias API de Servlets	Las acciones de S1 poseen dependencias sobre la API de Servlets desde que los objetos HttpServletRequest y HttpServletResponse son pasados como parámetros a los métodos de las acciones.	Las acciones de S2 no están acopladas al contenedor. La mayoría de los contextos de un servlet (Request, Session) son representados como simples Maps, permitiendo esto testear las acciones aisladamente.
Capacidad de testing	Uno de los mayores problemas de S1 es la dificultad de poder testear las acciones sin tener que levantar un servidor que las contenga. Esto es así dada la dependencia con la API de Servlets. Existen una extensión (Struts TestCase) que ofrece un conjunto de "mock objects" para poder realizar los tests.	Las acciones de S2 pueden ser testeadas directamente instanciando la Acción, asignarle valores a sus atributos e invocar el método que se quiere testear. El soporte de inyección de dependencias nos permite que el proceso de testing sea más sencillo aún.
Carga de formularios	S1 usa los objetos "ActionForm" para capturar la entrada en un formulario web. Tal como las acciones, todos los "forms" de nuestro sistema deben extender una clase base. Dado que no es posible usar JavaBeans como ActionForms es común que los desarrolladores escriban código redundante para capturar la entrada de los formularios web.	S2 utiliza las propiedades de una acción como "campos de entrada" de un formulario, con lo cual no es necesario utilizar objetos secundarios para capturar la entrada de los formularios. Particularmente es posible tener en la acción objetos de nuestro modelo o DTOs los cuales serán instanciados por S2 sin ser el desarrollador el que instancia los objetos y carga sus valores.
Lenguaje de expresiones	S1 esta integrado con JSTL, es decir que es posible hacer uso de JSTL EL (Expresión Lenguaje).	S2 puede usar JSTL pero provee soporte de un lenguaje de expresiones mucho más poderoso y flexible llamado "Object Graph Notation Language" (ONGL)
Conversión de tipos	Las propiedades de los formularios de S1 son generalmente simples "String". S1 utiliza "Commons-Beanutils" para la conversión de tipos. Las conversiones son por clase y no es posible configurarlo por instancia.	S2 utiliza ONGL para la conversión de tipos. El framework incluye conversiones para objetos y tipos primitivos.
Control de la ejecución de una acción	S1 soporta "request processors" por cada módulo, pero todas las acciones del módulo deben compartir el mismo ciclo de vida.	S2 permite crear diferentes ciclos de vida, uno por acción, mediante el uso de la pila de interceptores. Las pilas de interceptores configurados pueden ser creadas y usadas con diferentes acciones.

En la siguiente figura se representa la arquitectura de Struts 2:



Una vez descritas las características de Struts 2, en los apartados 4.4.2, 4.4.3 y 4.4.4 se hace referencia a cada una de las partes integrantes del patrón MVC, detallando para cada una su papel en la aplicación a implementar.

La solución propuesta para emplear el patrón MVC en el desarrollo de la aplicación es el uso del framework de aplicaciones web ***Struts 2.***

4.4.2 Modelo

El modelo de la aplicación a desarrollar hace referencia a los datos que maneja la aplicación y las reglas de negocio que operan sobre ellos. Esto en Struts 2 es el conjunto de acciones.

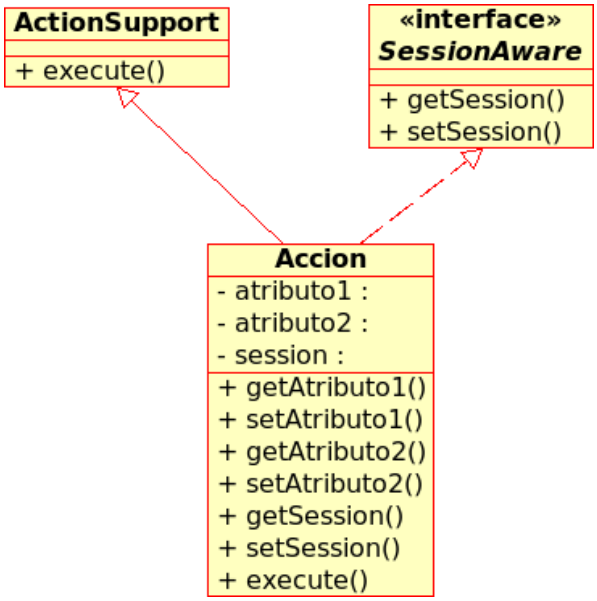
Las acciones se representarán en la aplicación a través de clases Java. Estas clases se caracterizan por ser clases simples que contienen una serie de atributos. Sin embargo, es necesario que tengan las siguientes características:

- Poseer la funcionalidad que propiamente las convierte en acciones para funcionar con Struts. Esto se consigue haciendo que cada una de las clases que van a funcionar como

acciones del modelo hereden de la clase abstracta ***ActionSupport***, clase proporcionada por Struts2 que contiene el método ***execute()***, que es donde se va a implementar la lógica de negocio de la acción.

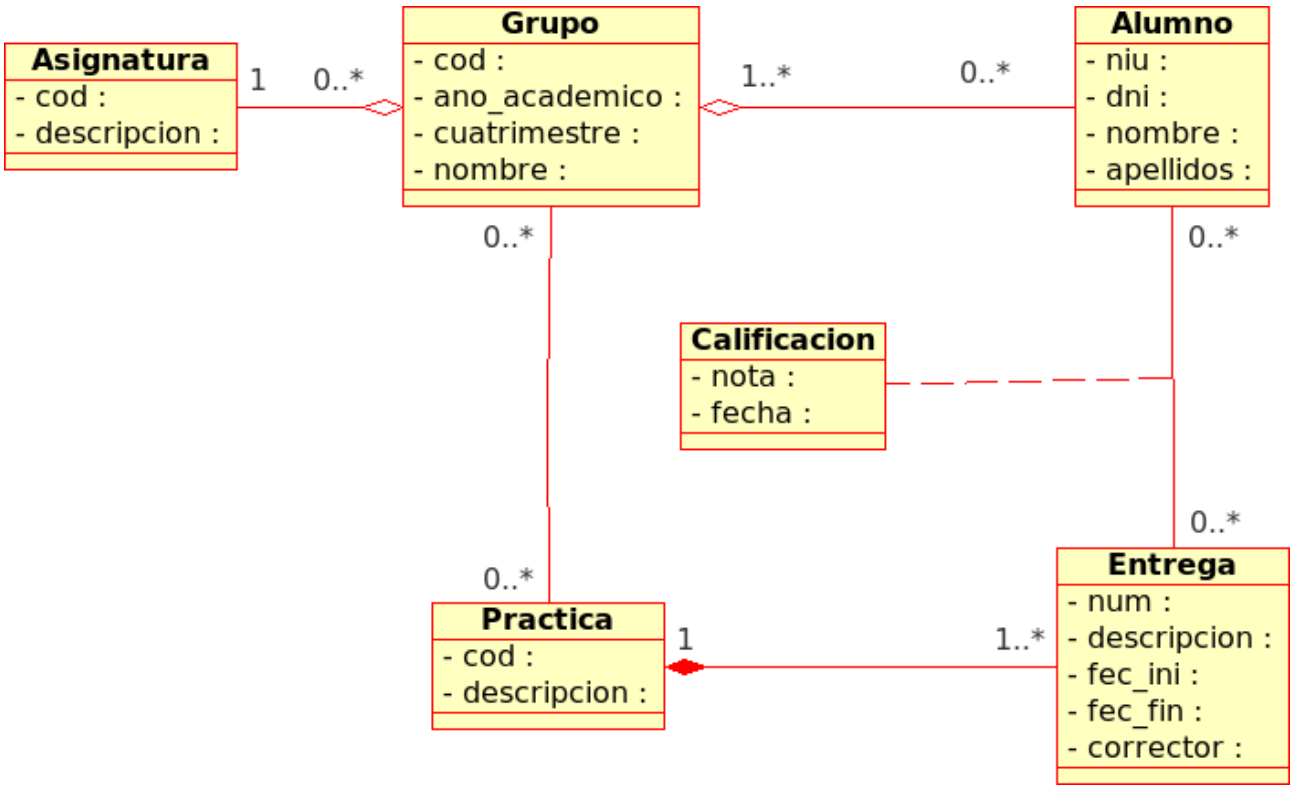
- Ser capaces de trabajar con una sesión de usuario. Esto nos va a permitir que únicamente se ejecuten las acciones en el caso de que el usuario haya abierto una sesión válida con el sistema. La forma de adquirir este comportamiento es que cada una de las clases (acciones) implemente el interfaz ***SessionAware***.

En la siguiente figura se representa el diagrama de clases de una acción genérica:



Hay que señalar que el conjunto de acciones a definir deben trabajar con el conjunto de clases que definen el dominio de la aplicación. Estas clases se han definido en el diagrama del apartado 3.4.4. y constituyen el conjunto de entidades de datos de la aplicación.

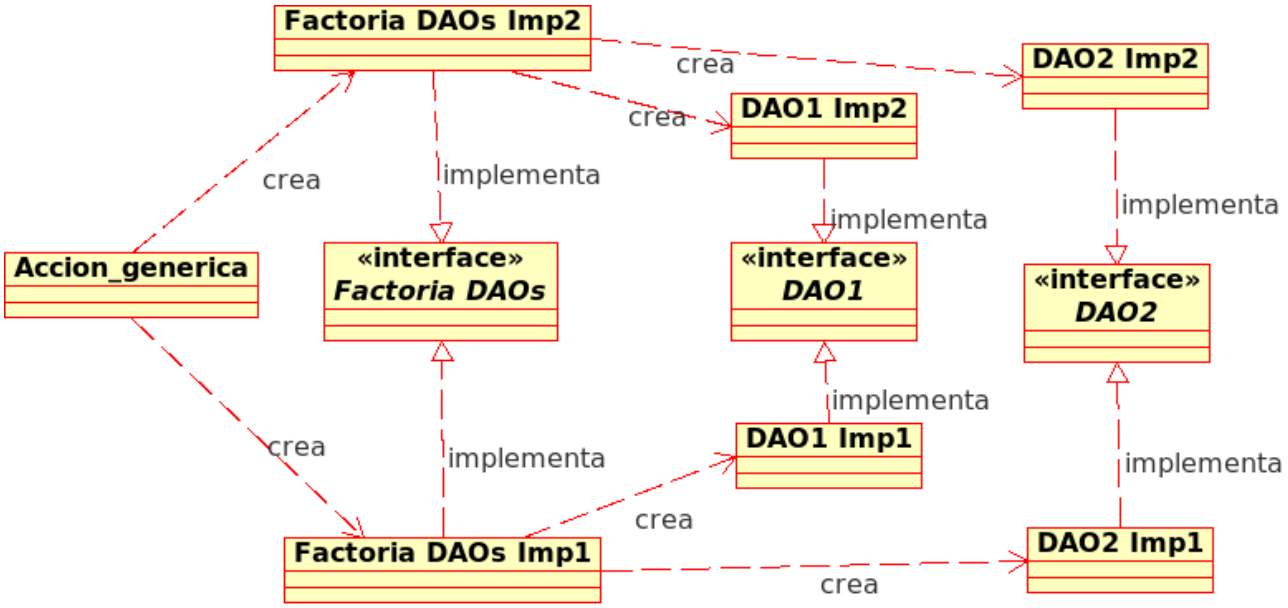
En este punto se vuelve a representar el diagrama de clases:



Llegado este punto, constituye una buena práctica de diseño el aplicar el patrón Data Access Object (DAO) a dichas entidades de datos. Esto consiste en que el conjunto de acciones del sistema trabajen con clases genéricas y no se preocupen de la implementación de las mismas, ya que en un futuro se puede tomar la decisión de cambiar la fuente de datos pero dejando intacta la lógica de negocio.

De manera lógica, los DAO's deben obtenerse de la misma fuente, por lo tanto se empleará el patrón de creación Factory Method, que nos permite contar con distintas factorías de DAOs que obedezcan a distintas fuentes de datos. Sin embargo, las acciones trabajarán con una factoría y unos DAOs abstractos. La fuente de datos a utilizar se definirá a partir de un fichero de configuración.

En el siguiente diagrama se representa la factoría de DAOs y su relación con una acción genérica:



4.4.3 Vista

La parte de la vista en el presente proyecto está compuesta por el conjunto de páginas jsp creadas para mostrar el estado del sistema en un momento dado. Para la creación de dichas páginas deben tenerse en cuenta los siguientes puntos:

- Requisitos de interfaz de usuario: deben cumplir los requisitos definidos en el apartado 3.3.2 del presente documento.
- Hoja de estilos: todas las páginas deberán presentar un aspecto similar que deben tomar de una hoja de estilos común.
- Etiquetas de Struts 2: las páginas deben hacer uso de las etiquetas de Struts 2 para manipular las distintas entidades. Para ello deben incluir la siguiente directiva en la cabecera: <%@ taglib uri="/struts-tags" prefix="s" %> . De esta forma se evitará la introducción de código java en la parte de presentación.

Ciertas páginas necesitan cambiar su contenido sin necesidad de recargar la página completa, cargando únicamente una parte concreta de la misma. Esta necesidad se identifica en las páginas en las que cierta información debe ser mostrada en función de la selección de un elemento determinado, concretamente un desplegable de opciones.

Para resolver este problema se decide utilizar la tecnología AJAX (Asynchronous Javascript And Xml). De esta forma, cada vez que se seleccione un elemento determinado de un desplegable de una página jsp se ejecutará una función en javascript que a su vez hará una llamada a una acción determinada que le proporcionará los datos necesarios para rellenar el elemento en cuestión.

4.4.4 Controlador

En este apartado se detalla el elemento controlador del patrón MVC en el marco de la aplicación a desarrollar.

El controlador de Struts 2 consiste en un expedidor de filtros (filter dispatcher), que contendrá la lógica de la aplicación en el fichero struts.xml.

El fichero struts.xml contiene:

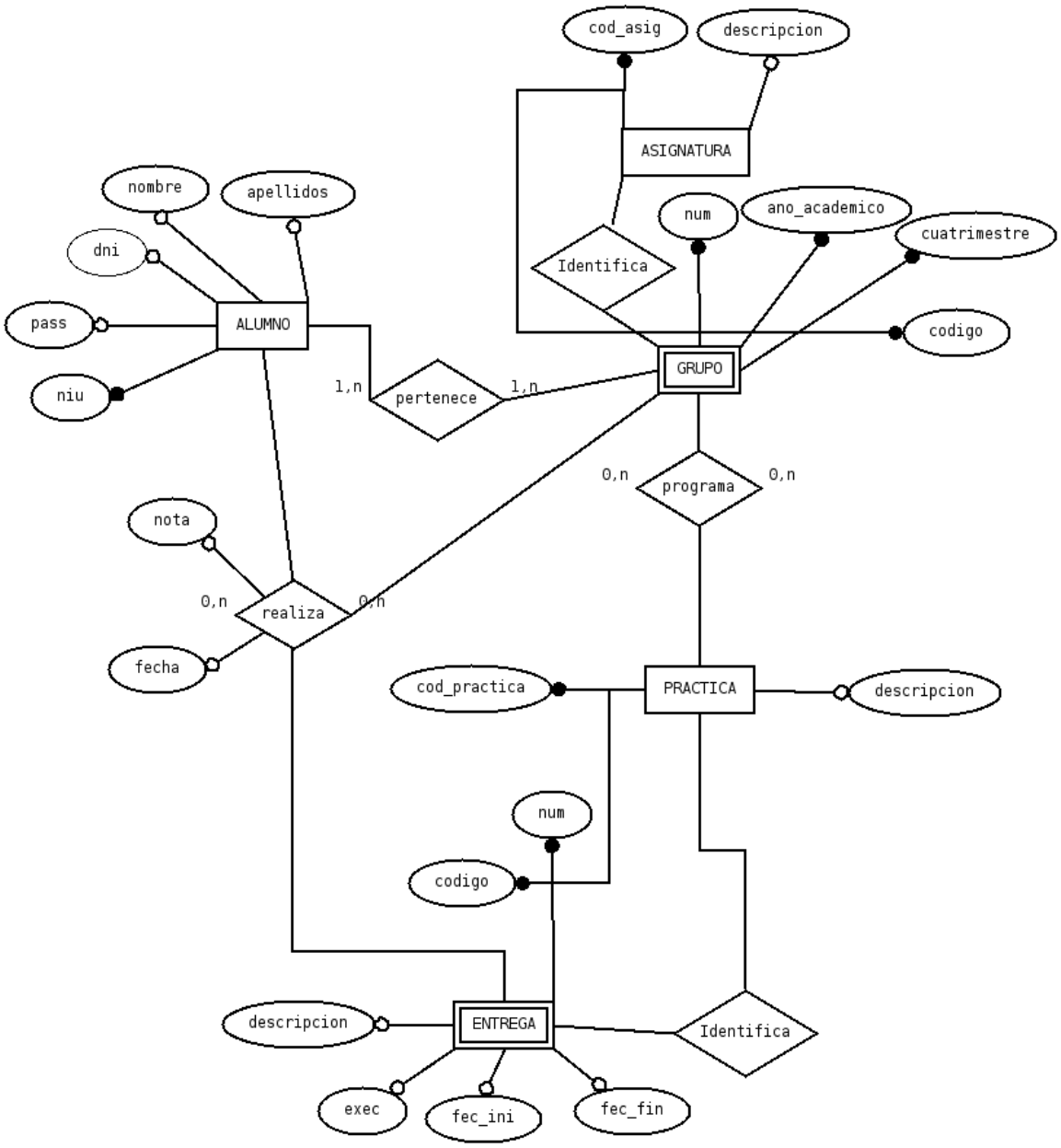
- El conjunto de posibles llamadas a la aplicación web por parte del usuario.
- Las acciones que se ejecutarán en cada una de las llamadas.
- Las páginas que se mostrarán en función del resultado del flujo de la acción a ejecutar.

Este fichero puede dividirse en distintos ficheros independientes, por lo que para una mejor estructuración y compresión del flujo de la aplicación se ha tomado la decisión de crear dos ficheros : uno para el conjunto de acciones del administrador y otro que recoja las acciones a ejecutar por el alumno.

4.5 Base de datos

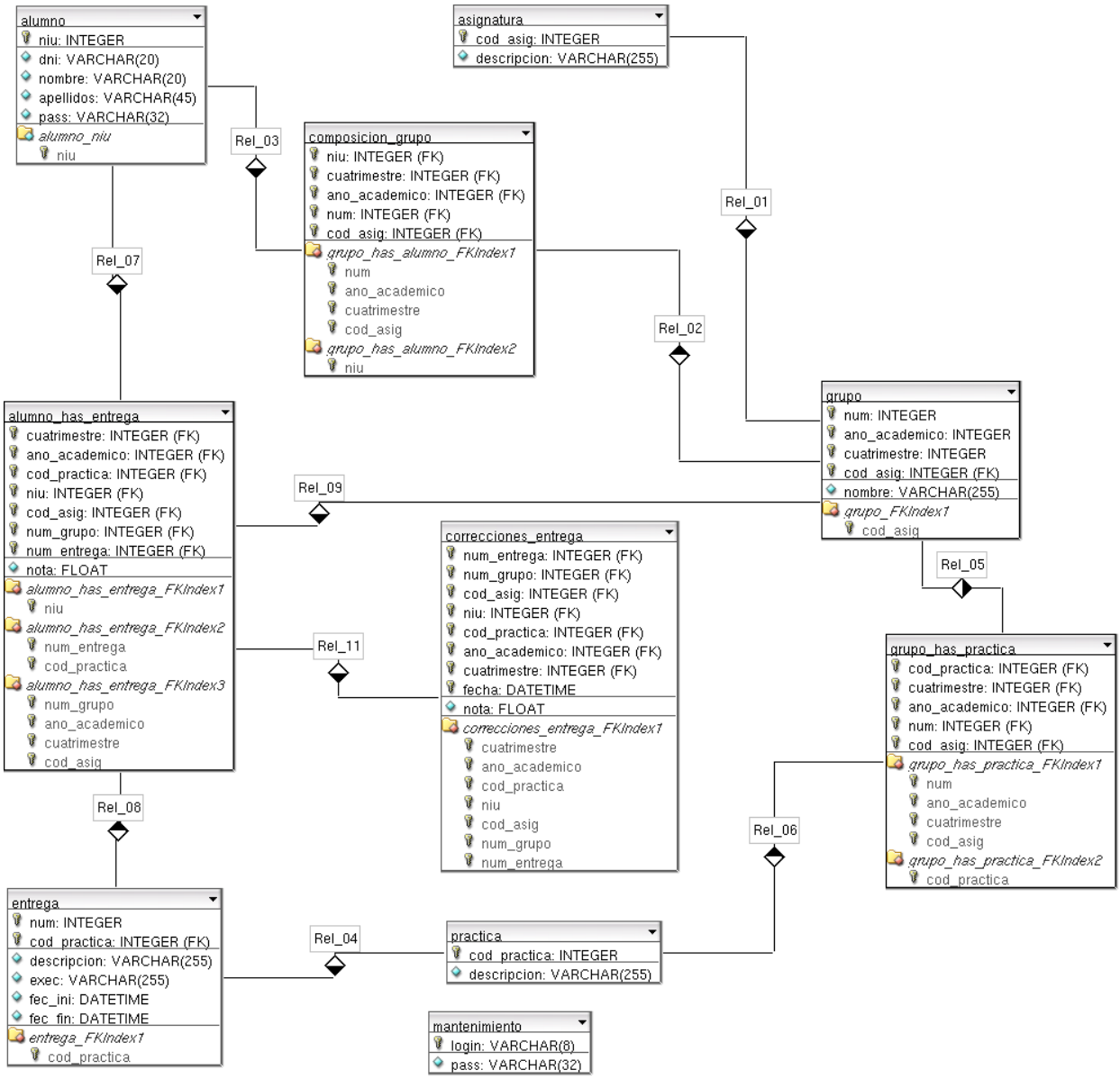
4.5.1 Diagrama Entidad/Relación

En este punto se define el diagrama Entidad/Relación, elemento fundamental en todo proceso de modelado de bases de datos. Permite describir una base de datos en el plano conceptual identificando los objetos participantes en la base de datos como relaciones, las cuales se vinculan entre ellas mediante relaciones.


cod_asig
descripcion
ASIGNATURA
nombre
apellidos
dni
pass
niu
ALUMNO
num
ano_academico
cuatrimestre
Identifica
codigo
GRUPO
1,n
pertenece
1,n
0,n
programa
0,n
nota
0,n
realiza
0,n
fecha
cod_practica
PRACTICA
descripcion
num
codigo
descripcion
ENTREGA
Identifica
exec
fec_ini
fec_fin

4.5.2 Modelo relacional

En este apartado se presenta el modelo relacional. Para ello se ha utilizado la herramienta de modelado de bases de datos relacionales DBDesigner, principalmente enfocada a MySQL.



CAPÍTULO 5:

IMPLEMENTACIÓN

5 Implementación

El objetivo de esta etapa es desarrollar la implementación de una aplicación software pueda ser ejecutada correctamente cumpliendo con los criterios especificados en las fases de análisis y diseño. La dificultad reside en cómo realizar esta implementación de la mejor manera posible, ya que va a depender de factores como el lenguaje de programación, la metodología empleada y el entorno de desarrollo establecido.

En este apartado se describirán los aspectos más relevantes del proceso de implementación, como el entorno de desarrollo utilizado, la estructuración del código fuente y la lógica de la aplicación.

5.1 Entorno de desarrollo

Para la codificación de la aplicación se tomó la decisión de emplear el entorno de desarrollo integrado Eclipse versión 3.4 (Ganymede). Constituye una plataforma muy potente para el desarrollo de proyectos Java, aunque permite programar en otros lenguajes tales como C, C++, Cobol, Python, Perl, PHP, etc.

Eclipse fue desarrollado originalmente por IBM, aunque actualmente es mantenido por la fundación Eclipse, una organización sin ánimo de lucro que fomenta el código abierto. Actualmente tiene una gran comunidad de desarrolladores y usuarios, por lo que existen grandes cantidades de documentación, foros y páginas webs especializadas. Su alta modularidad permite utilizar nuevas funcionalidades añadiendo distintos plugins que incrementan su ya de por sí gran potencial.

Los elementos externos que se han añadido a Eclipse para poder acometer la codificación del proyecto son los siguientes:

- Java Development Kit (JDK) versión 1.6.0_07. Contiene la máquina virtual Java necesaria para la ejecución del servidor de aplicaciones. Eclipse permite trabajar con distintas versiones máquinas virtuales, por lo que está asegurada la compatibilidad y

funcionamiento de cualquier elemento Java.

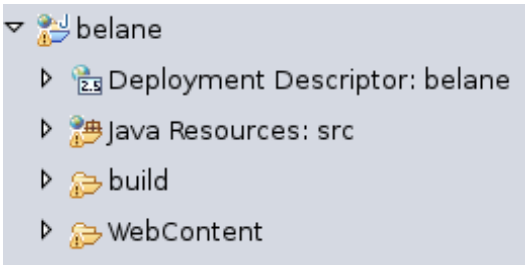
- Apache Tomcat 6.0.18 Es el servidor de aplicaciones que contiene la aplicación desarrollada. Eclipse también permite trabajar con distintos servidores de aplicaciones de forma totalmente integrada. Esto proporciona un entorno de desarrollo limpio, rápido y eficiente.

A partir de este punto, se trata de crear un proyecto nuevo en Eclipse, que será de la tipología "proyecto web dinámico". Dicho proyecto será desplegado en el servidor tomcat que se ha integrado en el entorno. De esta forma, el mismo Eclipse se ocupará de volver a desplegar la aplicación siempre que detecte algún cambio en la misma. Así no hay necesidad de compilar las distintas clases Java por separado.

5.2 Estructura del código

En este punto se detalla cómo se ha estructurado el código fuente de la aplicación. La estructura básica está determinada por las opciones por defecto que impone Eclipse a la hora de crear un proyecto web dinámico.

En la siguiente figura podemos ver la estructura del directorio raíz del proyecto:



En esta estructura encontramos los siguientes directorios:

- *belane*: Descriptor de despliegue de la aplicación.
- *Src*: Directorio que contiene el código fuente.
- *Build*: Directorio donde se almacenarán las clase compiladas y los distintos ficheros de configuración y recursos.

- *WebContent*: Contiene los elementos web de la aplicación como las páginas jsp, imágenes y hoja de estilos. También aloja las librerías de terceros que va a emplear el sistema.

A continuación se detalla el contenido de los directorios src y WebContent, por contener los elementos más relevantes.

5.2.1 Directorio src

Como se ha señalado anteriormente, este directorio contiene las distintas clases Java estructuradas en varios paquetes:

actions

En este paquete se alojan las acciones que definen el flujo de la aplicación. Dentro de él se han establecido dos paquetes importantes:

- *actions.alum*: Contiene las acciones referidas al usuario alumno.
- *actions.admin*: Aloja las acciones referidas al usuario administrador. Dado el número elevado de ellas se han agrupado en cuatro paquetes principales:
  - *actions.admin.asig*: Acciones relativas al tratamiento de asignaturas.
  - *actions.admin.group*: Acciones relacionadas con la gestión de grupos.
  - *actions.admin.pract*: Acciones que relacionadas con el tratamiento de prácticas.
  - *actions.admin.reports*: Acciones que contemplan la ejecución de los distintos informes de la aplicación.

También se alojan en el paquete dos ficheros xml (“admin.xml” y “alumno.xml”), cuyo contenido será detallado en el apartado 5.3 del presente capítulo.

conf

Este paquete contiene el fichero “belane.properties”, que es fichero principal de configuración de la aplicación. Su contenido se detalla en el apartado 7.4.2 del presente documento.

daos

Este paquete contiene las entidades de datos, que emplean el patrón de diseño Data Access Object, con las que va a trabajar la aplicación. Existen dos tipos:

- *interfaces*: Interfaces Java que van a implementar las distintas entidades de datos.
- *Implementaciones*: Clases Java que implementan los interfaces de las entidades de datos.

exceptions

Aquí se almacenan las distintas excepciones que puede generar la aplicación.

grades

Paquete que contiene las clases relativas a la corrección de los ejercicios entregados por los alumnos. Implementan la lógica de integración con los servicios web que ejecutan la función correctora.

reports

Almacena los informes Jasper que compila la aplicación cuando muestra un informe determinado.

util

Contiene conjuntos de clases de distinta índole utilizadas para apoyar las funcionalidades de la aplicación:

- *util.access*: Conjunto de clases relacionadas con el acceso y establecimiento de una sesión en la aplicación.
- *util.com*: Clases que implementan distintas funcionalidades orientadas a la red como el envío de correos electrónicos.
- *util.db*: Paquete que contiene el conjunto de clases necesarias para interactuar con la base de datos. Asimismo contiene el conjunto de sentencias SQL a emplear contra la base de datos.

- *util.file*: Utilidades para manipular ficheros.
- *util.xml*: Paquete que permite el tratamiento de ficheros xml.

5.2.2 Directorio WebContent

En este directorio se encuentran los siguientes elementos:

- ***img***: Contiene las imágenes estáticas de la parte web.
- ***META-INF***: Alberga el fichero "context.xml", donde se define la configuración que permite a la aplicación acceder a la base de datos.
- ***WEB-INF***: Aloja tres elementos diferenciados:

- *jsp*: Directorio que contiene el conjunto de páginas jsp que conforman la vista de la aplicación. La razón de alojarlo dentro del directorio WEB-INF obedece a la propia seguridad de la aplicación, ya que teniendo esta ubicación nos aseguramos que las páginas jsp no puedan ser accedidas directamente desde el navegador.

- *lib*: Directorio que aloja el conjunto de librerías de terceros necesarias para el funcionamiento de la aplicación.

- *web.xml*: Fichero descriptor de despliegue.

5.3 Lógica del controlador

La lógica del controlador se define en el fichero "struts.xml", que se encuentra alojado en el directorio src. Para una mejor organización se tomó la decisión de hacer dos referencias dentro de este fichero a otros dos xml ("admin.xml" y "alumno.xml") que contuvieran respectivamente lógica del controlador referida al flujo generado por el administrador por una parte y por otra la lógica que gobierna el flujo generado por el alumno.

Los ficheros "admin.xml" y "alumno.xml" se alojan en el paquete actions del directorio src de la aplicación. Con todo esto se detalla el contenido de los tres ficheros:

struts.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
```

"-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"

"http://struts.apache.org/dtds/struts-2.0.dtd">

<struts>

<include file="actions/admin.xml"/>

<include file="actions/alumno.xml"/>

</struts>

admin.xml

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE struts PUBLIC

"-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"

"http://struts.apache.org/dtds/struts-2.0.dtd">

<struts>

<package name="admin" extends="struts-default">

<action name="LoginAdmin" class="actions.admin.LoginAdminAction">

<result name="success">WEB-INF/jsp/menuAdm.jsp</result>

<result name="failure">index_admin.jsp</result>

</action>

<action name="LogoutAdmin" class="actions.alum.LogoutAlumAction">

<result name="success">index_admin.jsp</result>

<result name="failure">index_admin.jsp</result>

</action>

<action name="ToChangeAdminPassword"

class="actions.admin.GoChangeAdminPasswordAction">

<result name="success">WEB-INF/jsp/changePasswordAdmin.jsp</result>

<result name="failure">WEB-INF/jsp/error.jsp</result>

<result name="logout" type="redirectAction">LogoutAdmin</result>

</action>

<action name="ChangePasswordAdmin" class="actions.admin.ChangeAdminPasswordAction">

<result name="success">WEB-INF/jsp/menuAdm.jsp</result>

<result name="failure">WEB-INF/jsp/error.jsp</result>

<result name="logout" type="redirectAction">LogoutAdmin</result>

</action>

<action name="ToMenuAsig" class="actions.admin.asig.GoMenuAsig">

<result name="success">WEB-INF/jsp/menuAsig.jsp</result>

<result name="failure">WEB-INF/jsp/error.jsp</result>

<result name="logout" type="redirectAction">LogoutAdmin</result>

</action>

<action name="ToAddAsig" class="actions.admin.asig.GoAddAsigAction">

<result name="success">WEB-INF/jsp/addAsig.jsp</result>

<result name="failure">WEB-INF/jsp/error.jsp</result>

<result name="logout" type="redirectAction">LogoutAdmin</result>

</action>

<action name="AddAsig" class="actions.admin.asig.AddAsigAction">

<result name="success">WEB-INF/jsp/menuAsig.jsp</result>

<result name="failure">WEB-INF/jsp/error.jsp</result>

<result name="logout" type="redirectAction">LogoutAdmin</result>

</action>

<action name="ToDelAsig" class="actions.admin.asig.GoDelAsigAction">

<result name="success">WEB-INF/jsp/delAsig.jsp</result>

<result name="failure">WEB-INF/jsp/error.jsp</result>

<result name="logout" type="redirectAction">LogoutAdmin</result>

</action>

<action name="DelAsig" class="actions.admin.asig.DelAsigAction">

<result name="success">WEB-INF/jsp/menuAsig.jsp</result>

<result name="failure">WEB-INF/jsp/error.jsp</result>

<result name="logout" type="redirectAction">LogoutAdmin</result>

</action>

<action name="ToMenuGrupos" class="actions.admin.group.GoMenuGrupos">

<result name="success">WEB-INF/jsp/menuGrupos.jsp</result>

<result name="failure">WEB-INF/jsp/error.jsp</result>

<result name="logout" type="redirectAction">LogoutAdmin</result>

```
</action>
<action name="ToCreateGroup" class="actions.admin.group.GoCreateGroup">
<result name="success">WEB-INF/jsp/createGroup.jsp</result>
<result name="failure">WEB-INF/jsp/error.jsp</result>
<result name="logout" type="redirectAction">LogoutAdmin</result>
</action>
<action name="ToDeleteGroup" class="actions.admin.group.GoDeleteGroup">
<result name="success">WEB-INF/jsp/delGroup.jsp</result>
<result name="failure">WEB-INF/jsp/error.jsp</result>
<result name="logout" type="redirectAction">LogoutAdmin</result>
</action>
<action name="ToLoadAlumnosGroup" class="actions.admin.group.GoLoadAlumnosGroup">
<result name="success">WEB-INF/jsp/loadAlumnosGroup.jsp</result>
<result name="failure">WEB-INF/jsp/error.jsp</result>
<result name="logout" type="redirectAction">LogoutAdmin</result>
</action>
<action name="ToUnLoadAlumnosGroup"
class="actions.admin.group.GoUnloadAlumnosGroup">
<result name="success">WEB-INF/jsp/unloadAlumnosGroup.jsp</result>
<result name="failure">WEB-INF/jsp/error.jsp</result>
<result name="logout" type="redirectAction">LogoutAdmin</result>
</action>
<action name="ToDeleteAlumno" class="actions.admin.group.GoUnloadAlumnosGroup">
<result name="success">WEB-INF/jsp/deleteAlumno.jsp</result>
<result name="failure">WEB-INF/jsp/error.jsp</result>
<result name="logout" type="redirectAction">LogoutAdmin</result>
</action>
<action name="DeleteAlumno" class="actions.admin.group.DelAlumnoAction">
<result name="success">WEB-INF/jsp/menuGrupos.jsp</result>
<result name="failure">WEB-INF/jsp/error.jsp</result>
<result name="logout" type="redirectAction">LogoutAdmin</result>
```

</action>

<action name="DeleteGroup" class="actions.admin.group.DelGroupAction">

<result name="success">WEB-INF/jsp/menuGrupos.jsp</result>

<result name="failure">WEB-INF/jsp/error.jsp</result>

<result name="logout" type="redirectAction">LogoutAdmin</result>

</action>

<action name="UploadAlumnosGroup" class="actions.admin.group.LoadAlumnosGroupAction">

<result name="success">WEB-INF/jsp/menuGrupos.jsp</result>

<result name="failure">WEB-INF/jsp/error.jsp</result>

<result name="logout" type="redirectAction">LogoutAdmin</result>

</action>

<action name="UnloadAlumnosGroup"

class="actions.admin.group.UnloadAlumnosGroupAction">

<result name="success">WEB-INF/jsp/menuGrupos.jsp</result>

<result name="failure">WEB-INF/jsp/error.jsp</result>

<result name="logout" type="redirectAction">LogoutAdmin</result>

</action>

<action name="CreateGroup" class="actions.admin.group.CreateGroupAction">

<result name="success">WEB-INF/jsp/menuGrupos.jsp</result>

<result name="failure">WEB-INF/jsp/error.jsp</result>

<result name="logout" type="redirectAction">LogoutAdmin</result>

</action>

<action name="ToMenuPracticas" class="actions.admin.pract.GoMenuPracticas">

<result name="success">WEB-INF/jsp/menuPracticas.jsp</result>

<result name="failure">WEB-INF/jsp/error.jsp</result>

<result name="logout" type="redirectAction">LogoutAdmin</result>

</action>

<action name="ToAddPractica" class="actions.admin.pract.GoAddPracticaAction">

<result name="success">WEB-INF/jsp/addPractica.jsp</result>

<result name="failure">WEB-INF/jsp/error.jsp</result>

<result name="logout" type="redirectAction">LogoutAdmin</result>

```
</action>
<action name="AddPractica" class="actions.admin.pract.AddPracticaAction">
<result name="success">WEB-INF/jsp/menuPracticas.jsp</result>
<result name="failure">WEB-INF/jsp/error.jsp</result>
<result name="logout" type="redirectAction">LogoutAdmin</result>
</action>
<action name="ToDelPractica" class="actions.admin.pract.GoDelPracticaAction">
<result name="success">WEB-INF/jsp/delPractica.jsp</result>
<result name="failure">WEB-INF/jsp/error.jsp</result>
<result name="logout" type="redirectAction">LogoutAdmin</result>
</action>
<action name="DelPractica" class="actions.admin.pract.DelPracticaAction">
<result name="success">WEB-INF/jsp/menuPracticas.jsp</result>
<result name="failure">WEB-INF/jsp/error.jsp</result>
<result name="logout" type="redirectAction">LogoutAdmin</result>
</action>
<action name="ToAddEntrega" class="actions.admin.pract.GoAddEntregaAction">
<result name="success">WEB-INF/jsp/addEntrega.jsp</result>
<result name="failure">WEB-INF/jsp/error.jsp</result>
<result name="logout" type="redirectAction">LogoutAdmin</result>
</action>
<action name="AddEntrega" class="actions.admin.pract.AddEntregaAction">
<result name="success">WEB-INF/jsp/menuPracticas.jsp</result>
<result name="failure">WEB-INF/jsp/error.jsp</result>
<result name="logout" type="redirectAction">LogoutAdmin</result>
</action>
<action name="ToDelEntrega" class="actions.admin.pract.GoDelEntregaAction">
<result name="success">WEB-INF/jsp/delEntrega.jsp</result>
<result name="failure">WEB-INF/jsp/error.jsp</result>
<result name="logout" type="redirectAction">LogoutAdmin</result>
</action>
```

```
<action name="DelEntrega" class="actions.admin.pract.DelEntregaAction">
<result name="success">WEB-INF/jsp/menuPracticas.jsp</result>
<result name="failure">WEB-INF/jsp/error.jsp</result>
<result name="logout" type="redirectAction">LogoutAdmin</result>
</action>
<action name="ToAddPracticaGrupo" class="actions.admin.pract.GoAddPracticaGrupoAction">
<result name="success">WEB-INF/jsp/addPracticaGrupo.jsp</result>
<result name="failure">WEB-INF/jsp/error.jsp</result>
<result name="logout" type="redirectAction">LogoutAdmin</result>
</action>
<action name="AddPracticaGrupo" class="actions.admin.pract.AddPracticaGrupoAction">
<result name="success">WEB-INF/jsp/menuGrupos.jsp</result>
<result name="failure">WEB-INF/jsp/error.jsp</result>
<result name="logout" type="redirectAction">LogoutAdmin</result>
</action>
<action name="ToDelPracticaGrupo" class="actions.admin.group.GoDelPracticaGrupoAction">
<result name="success">WEB-INF/jsp/unloadPracticaGroup.jsp</result>
<result name="failure">WEB-INF/jsp/error.jsp</result>
<result name="logout" type="redirectAction">LogoutAdmin</result>
</action>
<action name="DelPracticaGrupo" class="actions.admin.pract.DelPracticaGrupoAction">
<result name="success">WEB-INF/jsp/menuGrupos.jsp</result>
<result name="failure">WEB-INF/jsp/error.jsp</result>
<result name="logout" type="redirectAction">LogoutAdmin</result>
</action>
<action name="ToListados" class="actions.admin.pract.GoListadosEntregasAction">
<result name="success">WEB-INF/jsp/listadosEntregas.jsp</result>
<result name="failure">WEB-INF/jsp/error.jsp</result>
<result name="logout" type="redirectAction">LogoutAdmin</result>
</action>
<action name="ShowAlumnos" class="actions.admin.group.ShowAlumnosAction">
```

<result>WEB-INF/jsp/showAlumnos.jsp</result>
<result name="failure">WEB-INF/jsp/errorVoid.jsp</result>
<result name="logout" type="redirectAction">LogoutAdmin</result>
</action>
<action name="ShowAlumnosDelete" class="actions.admin.group.ShowAlumnosDeleteAction">
<result>WEB-INF/jsp/showAlumnosDelete.jsp</result>
<result name="failure">WEB-INF/jsp/errorVoid.jsp</result>
<result name="logout" type="redirectAction">LogoutAdmin</result>
</action>
<action name="ShowPracticas" class="actions.admin.group.ShowPracticasAction">
<result>WEB-INF/jsp/showPracticas.jsp</result>
<result name="failure">WEB-INF/jsp/errorVoid.jsp</result>
<result name="logout" type="redirectAction">LogoutAdmin</result>
</action>
<action name="ShowEntregas" class="actions.admin.pract.ShowEntregasAction">
<result>WEB-INF/jsp/showEntregas.jsp</result>
<result name="failure">WEB-INF/jsp/errorVoid.jsp</result>
<result name="logout" type="redirectAction">LogoutAdmin</result>
</action>
<action name="ShowPracticasListados" class="actions.admin.group.ShowPracticasAction">
<result>WEB-INF/jsp/showPracticasListados.jsp</result>
<result name="logout" type="redirectAction">LogoutAdmin</result>
</action>
<action name="ListadoEntrega" class="actions.admin.reports.ShowNotasEntregaAction">
<result name="success">WEB-INF/jsp/listadosEntregas.jsp</result>
<result name="failure">WEB-INF/jsp/error.jsp</result>
<result name="logout" type="redirectAction">LogoutAdmin</result>
</action>
<action name="Test" class="actions.admin.TestWebServiceAction">
<result name="success">index_admin.jsp</result>
<result name="failure">WEB-INF/jsp/error.jsp</result>

```
<result name="logout" type="redirectAction">LogoutAdmin</result>
</action>
<action name="ToAddSchedule" class="actions.admin.pract.GoAddScheduleAction">
<result name="success">WEB-INF/jsp/addSchedule.jsp</result>
<result name="failure">WEB-INF/jsp/error.jsp</result>
<result name="logout" type="redirectAction">LogoutAdmin</result>
</action>
<action name="AddSchedule" class="actions.admin.pract.AddScheduleAction">
<result name="success">WEB-INF/jsp/menuPracticas.jsp</result>
<result name="failure">WEB-INF/jsp/error.jsp</result>
<result name="logout" type="redirectAction">LogoutAdmin</result>
</action>
</package>
</struts>
```

alumno.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
"-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
"http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
<package name="alumno" extends="struts-default">
<action name="LoginAlum" class="actions.alum.LoginAlumAction">
<result name="success">WEB-INF/jsp/menuAlum.jsp</result>
<result name="failure">index.jsp</result>
</action>
<action name="LogoutAlum" class="actions.alum.LogoutAlumAction">
<result name="success">index.jsp</result>
<result name="failure">index.jsp</result>
</action>
<action name="ChangeAlumPassword" class="actions.alum.ChangeAlumPasswordAction">
```

<result name="success">WEB-INF/jsp/menuAlum.jsp</result>
<result name="failure">WEB-INF/jsp/errorAlumno.jsp</result>
<result name="logout" type="redirectAction">Logout</result>
</action>
<action name="ToGeneratePassword" class="actions.alum.GoGeneratePasswordAction">
<result name="success">WEB-INF/jsp/generatePassword.jsp</result>
<result name="failure">WEB-INF/jsp/errorAlumno.jsp</result>
</action>
<action name="GeneratePassword" class="actions.alum.GeneratePasswordAction">
<result name="success">WEB-INF/jsp/generatePassword.jsp</result>
<result name="failure">index.jsp</result>
</action>
<action name="ToAlumData" class="actions.alum.GoAlumDataAction">
<result name="success">WEB-INF/jsp/menuAlumData.jsp</result>
<result name="failure">WEB-INF/jsp/errorAlumno.jsp</result>
<result name="logout" type="redirectAction">Logout</result>
</action>
<action name="ToChangePassword" class="actions.alum.GoChangePasswordAction">
<result name="success">WEB-INF/jsp/changePasswordAlum.jsp</result>
<result name="failure">WEB-INF/jsp/errorAlumno.jsp</result>
<result name="logout" type="redirectAction">Logout</result>
</action>
<action name="ToShowPracticas" class="actions.alum.GoShowPracticasAction">
<result name="success">WEB-INF/jsp/practicasAlum.jsp</result>
<result name="failure">WEB-INF/jsp/errorAlumno.jsp</result>
<result name="logout" type="redirectAction">Logout</result>
</action>
<action name="SelectEntrega" class="actions.alum.SelectEntregaAction">
<result name="success">WEB-INF/jsp/uploadEntrega.jsp</result>
<result name="failure">WEB-INF/jsp/errorAlumno.jsp</result>
<result name="logout" type="redirectAction">Logout</result>

</action>

<action name="UploadFileEntrega" class="actions.alum.UploadFileAction">

<result name="success">WEB-INF/jsp/menuAlum.jsp</result>

<result name="failure">WEB-INF/jsp/errorAlumno.jsp</result>

<result name="logout" type="redirectAction">Logout</result>

</action>

<action name="ToShowNotasPracticas" class="actions.alum.GoShowNotasPracticasAction">

<result name="success">WEB-INF/jsp/notasPracticasAlum.jsp</result>

<result name="failure">WEB-INF/jsp/errorAlumno.jsp</result>

<result name="logout" type="redirectAction">Logout</result>

</action>

<action name="ShowPracticasAlumno" class="actions.alum.ShowPracticasAction">

<result>WEB-INF/jsp/showPracticasAlumno.jsp</result>

<result name="failure">WEB-INF/jsp/errorVoid.jsp</result>

<result name="logout" type="redirectAction">LogoutAdmin</result>

</action>

<action name="ShowPracticasAlumnoNotas" class="actions.alum.ShowPracticasAction">

<result>WEB-INF/jsp/showPracticasAlumnoNotas.jsp</result>

<result name="failure">WEB-INF/jsp/errorVoid.jsp</result>

<result name="logout" type="redirectAction">LogoutAdmin</result>

</action>

<action name="ShowEntregasAlumno" class="actions.alum.ShowEntregasAlumnosAction">

<result>WEB-INF/jsp/showEntregasAlumnos.jsp</result>

<result name="failure">WEB-INF/jsp/errorVoid.jsp</result>

<result name="logout" type="redirectAction">LogoutAdmin</result>

</action>

<action name="ShowEntregasAlumnoNotas"

class="actions.alum.ShowEntregasAlumnosNotasAction">

<result>WEB-INF/jsp/showEntregasAlumnosNotas.jsp</result>

<result name="failure">WEB-INF/jsp/errorVoid.jsp</result>

<result name="logout" type="redirectAction">LogoutAdmin</result>

```
</action>
<action name="ShowNotaEntrega" class="actions.alum.ShowNotasEntregaAction">
<result name="success">WEB-INF/jsp/practicasAlum.jsp</result>
<result name="failure">WEB-INF/jsp/errorAlumno.jsp</result>
<result name="logout" type="redirectAction">Logout</result>
</action>
<action name="ShowPracticasAlumnoDef"
class="actions.alum.ShowPracticasAlumnoDefAction">
<result name="success">WEB-INF/jsp/practicasAlum2.jsp</result>
<result name="failure">WEB-INF/jsp/errorAlumno.jsp</result>
<result name="logout" type="redirectAction">Logout</result>
</action>
</package>
</struts>
```

5.4 Script de creación de la base de datos

A continuación se detalla el contenido del script que recrea las tablas integrantes de la base de datos:

```
CREATE TABLE alumno (
  niu INTEGER UNSIGNED NOT NULL,
  dni VARCHAR(20) NOT NULL,
  nombre VARCHAR(20) NULL,
  apellidos VARCHAR(45) NULL,
  pass VARCHAR(32) NULL,
  PRIMARY KEY(niu),
  UNIQUE INDEX alumno_niu(niu)
);

CREATE TABLE alumno_has_entrega (
  cuatrimestre INTEGER UNSIGNED NOT NULL,
```

```
  ano_academico INTEGER UNSIGNED NOT NULL,
  cod_practica INTEGER UNSIGNED NOT NULL,
  niu INTEGER UNSIGNED NOT NULL,
  cod_asig INTEGER UNSIGNED NOT NULL,
  num_grupo INTEGER UNSIGNED NOT NULL,
  num_entrega INTEGER UNSIGNED NOT NULL,
  nota FLOAT NULL,
    PRIMARY KEY(cuatrimestre, ano_academico, cod_practica, niu, cod_asig, num_grupo,
num_entrega),
  INDEX alumno_has_entrega_FKIndex1(niu),
  INDEX alumno_has_entrega_FKIndex2(num_entrega, cod_practica),
  INDEX alumno_has_entrega_FKIndex3(num_grupo, ano_academico, cuatrimestre, cod_asig)
);

CREATE TABLE asignatura (
  cod_asig INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  descripcion VARCHAR(255) NULL,
  PRIMARY KEY(cod_asig)
);

CREATE TABLE composicion_grupo (
  niu INTEGER UNSIGNED NOT NULL,
  cuatrimestre INTEGER UNSIGNED NOT NULL,
  ano_academico INTEGER UNSIGNED NOT NULL,
  num INTEGER UNSIGNED NOT NULL,
  cod_asig INTEGER UNSIGNED NOT NULL,
  PRIMARY KEY(niu, cuatrimestre, ano_academico, num, cod_asig),
  INDEX grupo_has_alumno_FKIndex1(num, ano_academico, cuatrimestre, cod_asig),
  INDEX grupo_has_alumno_FKIndex2(niu)
);
```

```
CREATE TABLE correcciones_entrega (
  num_entrega INTEGER UNSIGNED NOT NULL,
  num_grupo INTEGER UNSIGNED NOT NULL,
  cod_asig INTEGER UNSIGNED NOT NULL,
  niu INTEGER UNSIGNED NOT NULL,
  cod_practica INTEGER UNSIGNED NOT NULL,
  ano_academico INTEGER UNSIGNED NOT NULL,
  cuatrimestre INTEGER UNSIGNED NOT NULL,
  fecha DATETIME NOT NULL,
  nota FLOAT NULL,
    PRIMARY KEY(num_entrega, num_grupo, cod_asig, niu, cod_practica, ano_academico,
cuatrimestre, fecha),
    INDEX correcciones_entrega_FKIndex1(cuatrimestre, ano_academico, cod_practica, niu,
cod_asig, num_grupo, num_entrega)
);

CREATE TABLE entrega (
  num INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  cod_practica INTEGER UNSIGNED NOT NULL,
  descripcion VARCHAR(255) NULL,
  exec VARCHAR(255) NULL,
  fec_ini DATETIME NULL,
  fec_fin DATETIME NULL,
  PRIMARY KEY(num, cod_practica),
  INDEX entrega_FKIndex1(cod_practica)
);

CREATE TABLE grupo (
  num INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  ano_academico INTEGER UNSIGNED NOT NULL,
  cuatrimestre INTEGER UNSIGNED NOT NULL,
```

```
  cod_asig INTEGER UNSIGNED NOT NULL,
  nombre VARCHAR(255) NULL,
  PRIMARY KEY(num, ano_academico, cuatrimestre, cod_asig),
  INDEX grupo_FKIndex1(cod_asig)
);

CREATE TABLE grupo_has_practica (
  cod_practica INTEGER UNSIGNED NOT NULL,
  cuatrimestre INTEGER UNSIGNED NOT NULL,
  ano_academico INTEGER UNSIGNED NOT NULL,
  num INTEGER UNSIGNED NOT NULL,
  cod_asig INTEGER UNSIGNED NOT NULL,
  PRIMARY KEY(cod_practica, cuatrimestre, ano_academico, num, cod_asig),
  INDEX grupo_has_practica_FKIndex1(num, ano_academico, cuatrimestre, cod_asig),
  INDEX grupo_has_practica_FKIndex2(cod_practica)
);

CREATE TABLE mantenimiento (
  login VARCHAR(8) NOT NULL,
  pass VARCHAR(32) NULL,
  PRIMARY KEY(login)
);

CREATE TABLE practica (
  cod_practica INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  descripcion VARCHAR(255) NULL,
  PRIMARY KEY(cod_practica)
);
```

5.5 Distribución de la aplicación

En los apartados anteriores del presente capítulo se ha definido la estructura básica del

código de la aplicación. Sin embargo es necesario empaquetar esta estructura en un formato válido que se pueda desplegar en cualquier servidor de aplicaciones que contemple aplicaciones Java EE.

Para esto se utilizará la funcionalidad de Eclipse que permite exportar un proyecto en un archivo con formato .war. Este archivo será empleado en el despliegue en producción de la aplicación, y es incluido junto con el código fuente de la aplicación en un CD asociado al presente documento.

CAPÍTULO 6:

MANUAL DE USUARIO

6 Manual de usuario

El presente apartado contiene la documentación necesaria para utilizar eficazmente la aplicación desarrollada.

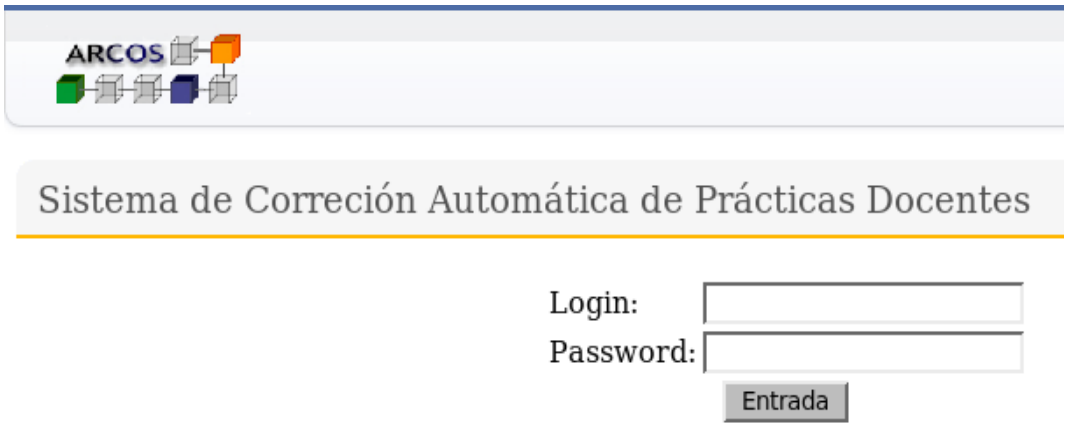
Esta documentación se establece tanto para el punto de vista del administrador como del alumno.

6.1 Documentación orientada al administrador

6.1.1 Acceso a la aplicación

El administrador accede a la aplicación introduciendo un identificador de usuario y una contraseña asociada. En caso de no ser correctos el sistema le informa de la situación y deniega el acceso a la aplicación.

Por defecto, inicialmente solo existe un usuario en el sistema y el nombre del mismo es “admin”.



6.1.2 Barra de menú principal

Esta barra de menú contiene la navegación principal de la aplicación, permitiendo acceder a las siguientes opciones:

- Cambio de contraseña.
- Menú de asignaturas.
- Menú de grupos.
- Menú de prácticas.
- Salida del sistema

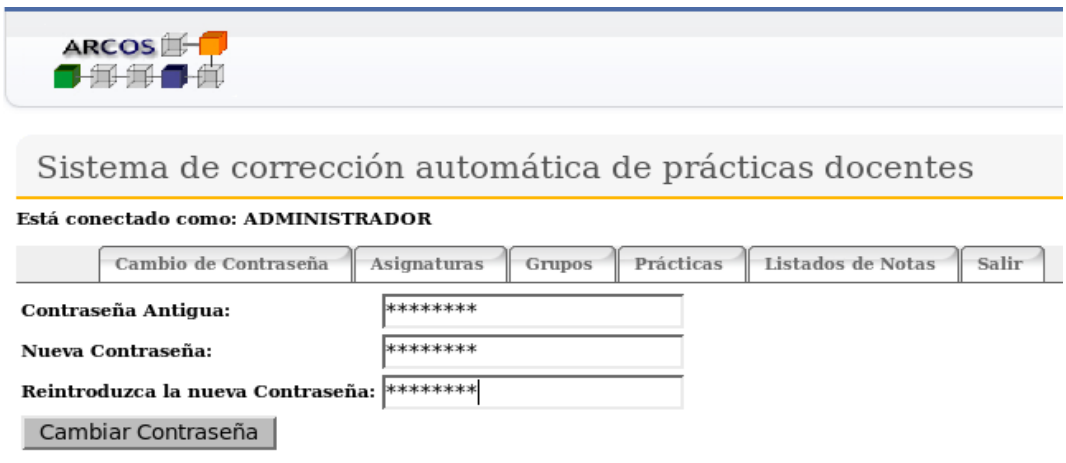
6.1.3 Cambio de contraseña

En esta opción el administrador tiene la posibilidad de cambiar la contraseña de acceso a la aplicación. Para ello se presentan tres campos: uno para introducir la nueva contraseña y otros dos que albergarán el nuevo valor.

Cabe señalar que para que se ejecute el cambio deben cumplirse dos condiciones:

- La contraseña actual introducida sea la correcta.
- La contraseña nueva debe ser exactamente igual en los dos campos habilitados para ello.

Una vez introducidos los datos se pulsará el botón con la leyenda “Cambiar Contraseña”.



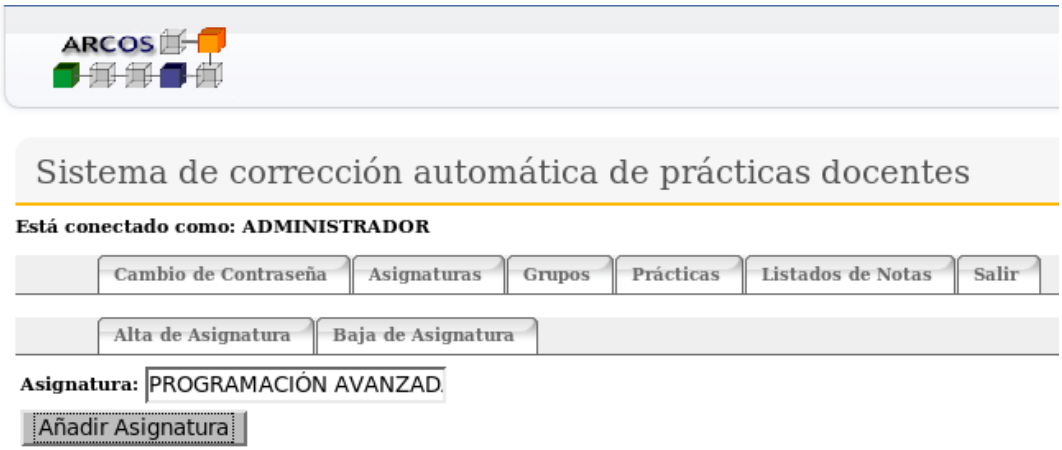
6.1.4 Menú de asignaturas

En esta parte de la aplicación se tratan todas las opciones relativas al tratamiento de asignaturas. Contiene las siguientes opciones:

- Alta de asignatura.
- Baja de asignatura.

6.1.5 Alta de asignatura

Se posibilita al administrador para que inserte una nueva asignatura en el sistema. La información necesaria que debe facilitar es el nombre identificativo de la asignatura.
Una vez introducido el nombre debe pulsar en el botón “Añadir Asignatura”. Si hay algún problema en el proceso, la aplicación lo notifica al usuario.

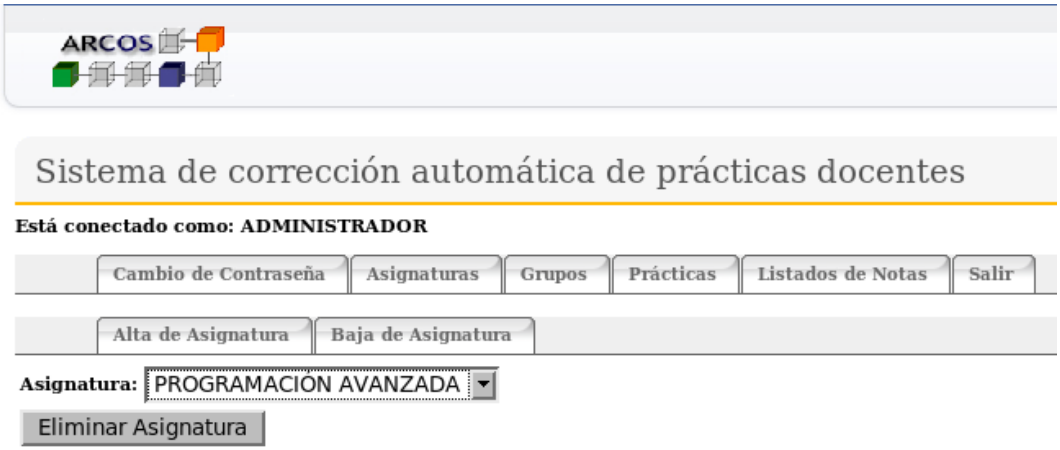


6.1.6 Baja de asignatura

En esta parte de la aplicación el administrador puede dar de baja una asignatura. Para ello se

le presentará un desplegable en pantalla con todas las asignaturas dadas de alta en la aplicación. El administrador deberá seleccionar la asignatura a dar de baja y pulsar el botón “Eliminar Asignatura”. En caso de que la aplicación detecte alguna condición por la que no pueda efectuarse el proceso de baja, se notificará al administrador por pantalla.

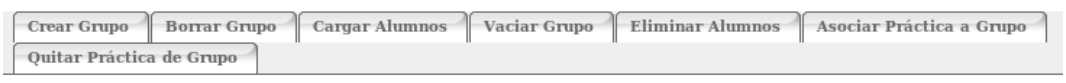
La condición determinante para poder eliminar una asignatura es que no exista un grupo creado de la misma.



6.1.7 Menú de grupos

Este menú de la aplicación permite acceder a las operaciones relacionadas con un grupo. Contiene las siguientes opciones:

- Creación de grupo.
- Borrado de grupo.
- Carga de alumnos al grupo.
- Eliminación de todos los alumnos del grupo.
- Eliminación individual de un alumno.
- Asociación de una práctica con un grupo.
- Eliminar la asociación de una práctica con un grupo.

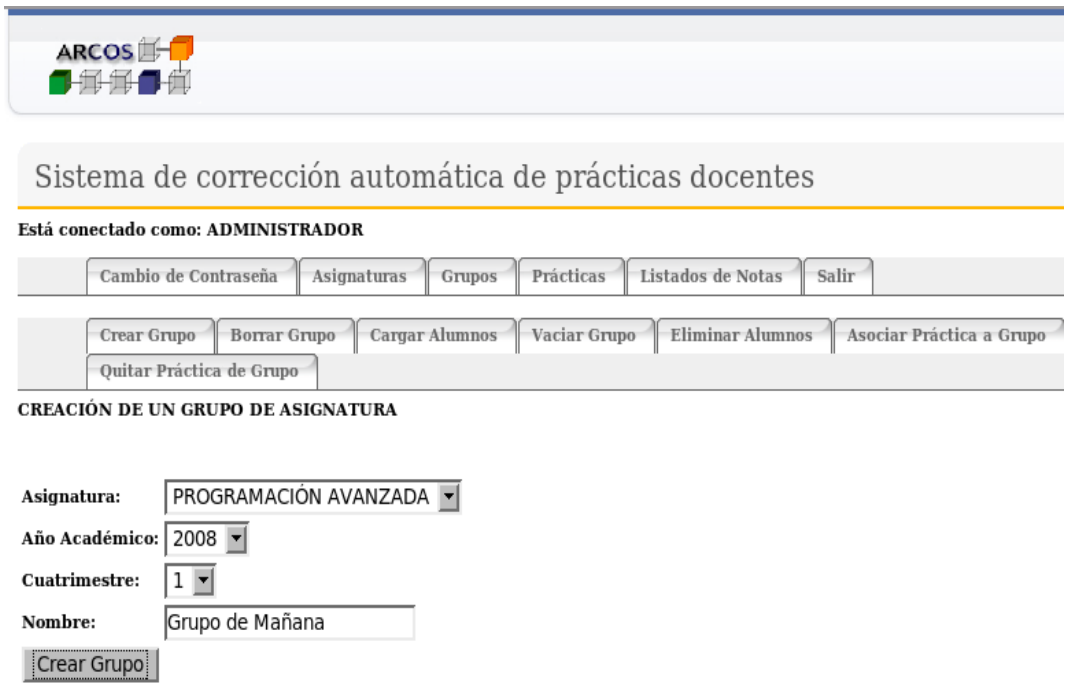


6.1.8 Creación de grupo

Esta opción permite la creación de un grupo. En este punto conviene recordar que un grupo se define por los siguientes atributos: asignatura, año académico, cuatrimestre y nombre.
Éstos serán los atributos que se nos permitirán introducir en la pantalla:

- Asignatura: se muestra un desplegable con todas las asignaturas dadas de alta en el sistema.
- Año académico: desplegable con el año académico configurado en la aplicación (ver el atributo del fichero "belane.properties" en el capítulo 7, donde se especifica la configuración y despliegue de la aplicación).
- Cuatrimestre: desplegable en el que se muestran los dos valores posibles para el cuatrimestre: primero y segundo.
- Nombre: campo de texto en el que se permite asignar un nombre identificativo para el grupo.

Una vez consignados todos los campos se pulsará en el botón "Crear Grupo". En caso de que exista alguna condición que no permita ejecutar el alta del grupo, será notificada por pantalla.

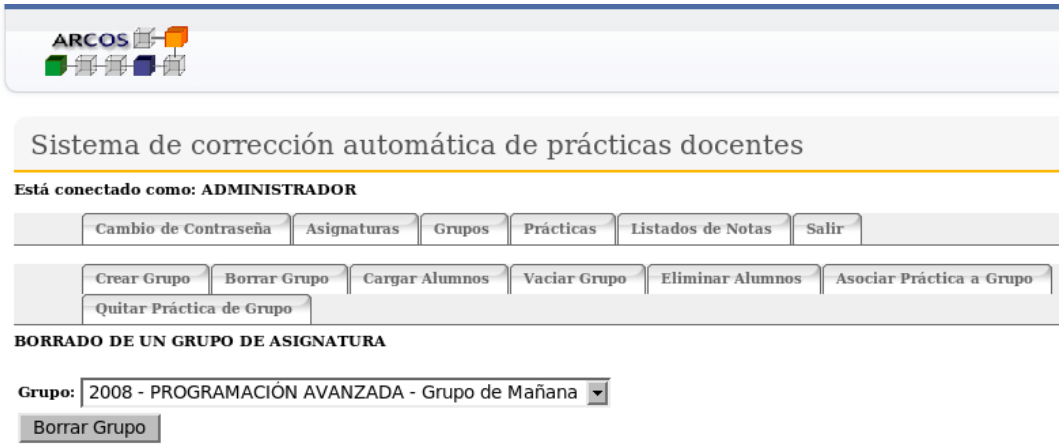


6.1.9 Baja de grupo

Esta opción permite la posibilidad de dar de baja un grupo del sistema. Para ello se nos presenta por pantalla un desplegable con todos los grupos creados en la aplicación. Para una correcta identificación de cada uno de ellos se presenta el año académico, la denominación de la asignatura y el nombre identificativo del grupo.

Una vez seleccionado el grupo será necesario pulsar en el botón con la leyenda “Borrar Grupo” y en caso de que exista algún problema, el sistema lo notificará por pantalla.
A continuación se definen las situaciones en las que no sería posible el borrado de un grupo:

- El grupo tiene alumnos asociados. Antes de proceder al borrado del grupo es necesario eliminar el conjunto de alumnos asociados al grupo (opción “Eliminar Alumnos” de la barra del menú de grupos).
- El grupo tiene prácticas asociadas, por lo que antes de borrar el grupo es necesario eliminar la asociación de las prácticas al grupo (menú “Quitar Practica de Grupo” de la barra de menú de Grupos).



6.1.10 Cargar Alumnos

Esta opción permite asociar un conjunto de alumnos a un grupo previamente creado.

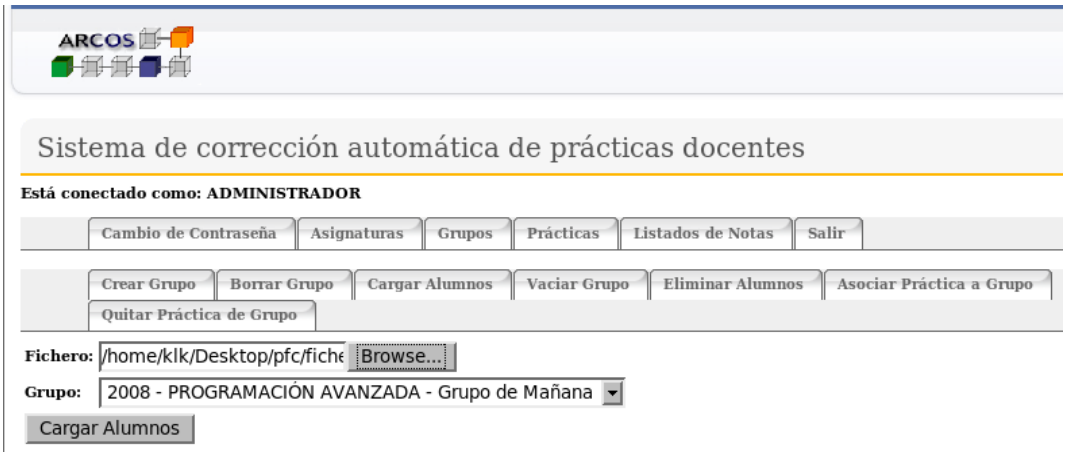
Para ello se presentará por pantalla un desplegable con el conjunto de los grupos creados en el sistema y un campo para poder cargar el archivo con el conjunto de alumnos que queremos incluir en el grupo. Como en otras opciones de la aplicación, el grupo está identificado por pantalla con los campos año académico, denominación de asignatura y nombre del grupo.

Para poder cargar el fichero con los alumnos pulsaremos el botón con la leyenda “Browse” y a continuación navegaremos por el sistema de archivos del equipo hasta seleccionar el fichero con los alumnos que queremos incorporar al grupo.

En este punto tenemos que recordar que el fichero con los alumnos debe obtenerse de los listados oficiales que se proporcionar a través del portal de la Universidad Carlos III de Madrid, ya que en ellos se encuentran los alumnos oficialmente matriculados.

Una vez seleccionados el fichero en formato excel con los alumnos y el grupo se pulsará el botón “Cargar Alumnos”, y en caso de existir algún error se mostrará por pantalla.

Conviene recordar que los ficheros con los listados oficiales de clase son dinámicos y que se generan a diario, por lo que sufren cambios. Para resolver esta situación podemos volver a cargar el fichero de alumnos actualizado, ya que la aplicación se encarga de añadir los alumnos que todavía no se encontraban en el grupo.



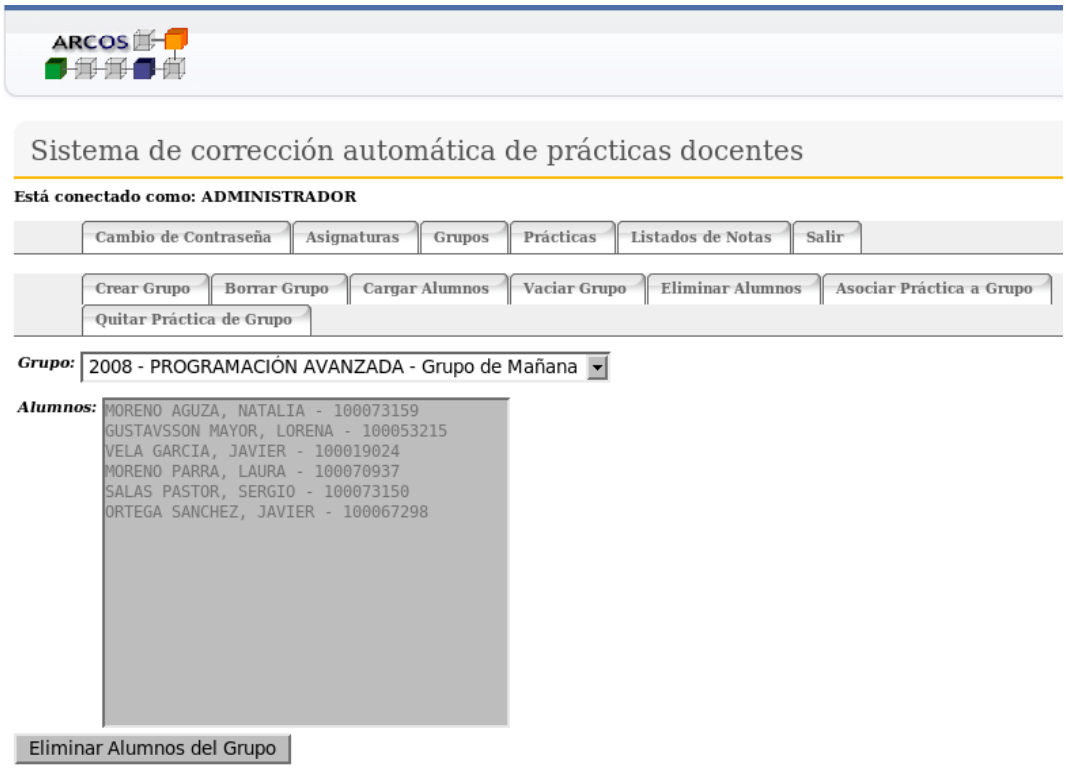
6.1.11 Vaciar grupo

En esta parte de la aplicación se permite eliminar el conjunto de alumnos de un grupo. Para ello se nos presenta un desplegable con los grupos presentes en el sistema y un cuadro de texto con los alumnos pertenecientes al mismo.

Para los grupos se presenta el año académico, la asignatura y el nombre identificativo, mientras que para los alumnos se muestra el nombre, los apellidos y el número de identificación de usuario (NIU) de la Universidad.

Una vez seleccionado el grupo en el que queremos eliminar los alumnos pulsaremos el botón "Eliminar Alumnos del Grupo", mostrando por pantalla algún mensaje en caso de que existiera algún problema.

Conviene recordar que esta operación solo será posible si un alumno del grupo ha realizado alguna entrega de una práctica.



6.1.12 Eliminar alumno

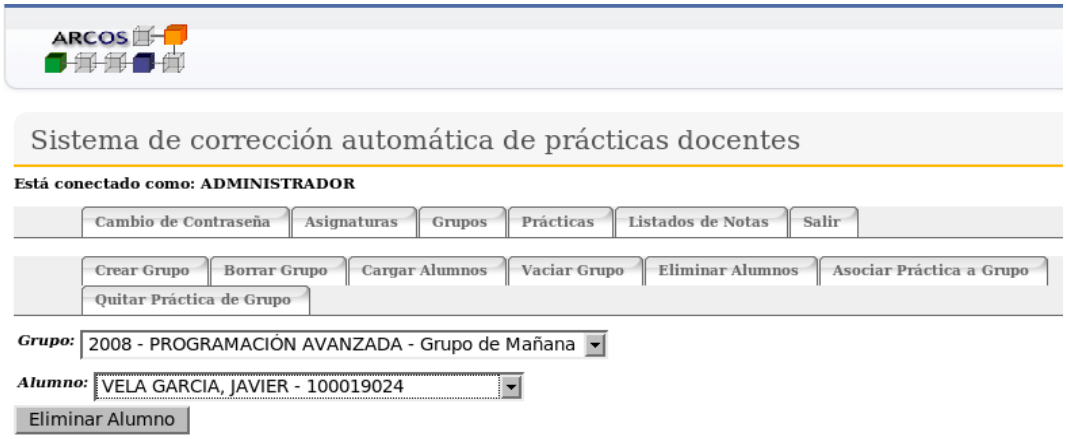
Esta opción del sistema permite dar de baja un alumno de un grupo determinado. Para ésto se presentan dos desplegables: uno con el conjunto de grupos existentes en el sistema y otro que contiene los alumnos pertenecientes al grupo seleccionado.

Cabe destacar que el desplegable de los alumnos se actualiza automáticamente cada vez que se selecciona un grupo nuevo.

Una vez seleccionado el grupo y el alumno en cuestión debe pincharse el botón “Eliminar Alumno”. En el caso de ocurrir algún error en el proceso la aplicación lo muestra por pantalla. El resultado de esta operación es que el alumno se borra del grupo seleccionado, y si además no se encuentra en ningún otro grupo, se elimina completamente el alumno, no pudiendo entrar de nuevo en el sistema.

A continuación se señalan las condiciones en las que no se permite borrar un alumno:

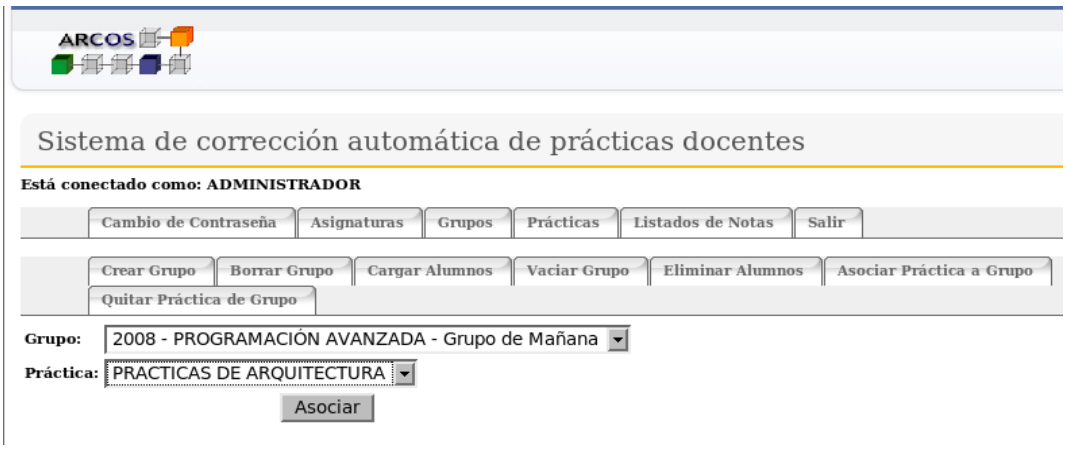
- El alumno ha entregado ficheros asociados a alguna práctica, a su vez asociada al grupo del que se le pretende borrar.
- El alumno tiene hechas correcciones de ficheros asociados a alguna práctica, a su vez asociada al grupo del que se le pretende borrar.



6.1.13 Asociar práctica a grupo

En esta parte del menú de grupos se ofrece la posibilidad de asociar una práctica previamente creada a un grupo. Esto implica que a partir de ese momento, los alumnos pertenecientes al grupo seleccionado podrán realizar entregas de ficheros asociadas al conjunto de entregas de las que se compone la práctica en cuestión.

Para esto se presentan dos desplegables por pantalla: uno con todos los grupos presentes en el sistema y otro con todas las prácticas creadas en el mismo. Una vez seleccionados el grupo y la práctica se pulsará el botón “Asociar”. En caso de que exista algún tipo de problema se mostrará por pantalla.

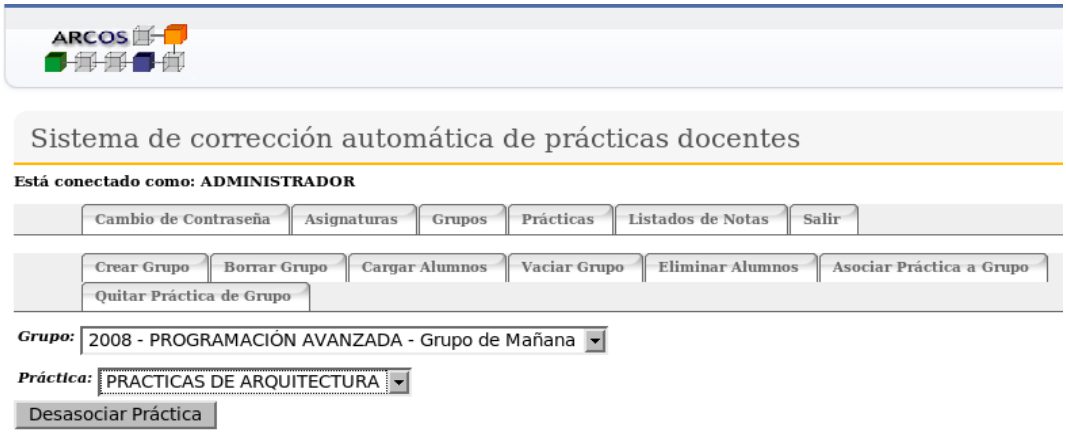


6.1.14 Quitar práctica de grupo

Esta opción permite que una práctica determinada deje de estar asociada a un grupo. Para ello se presentan dos desplegables, uno para el grupo y otro para la práctica, que nos mostrarán la información almacenada en el sistema.

Hay que destacar que cada vez que seleccionemos un grupo en su desplegable la aplicación refresca el contenido del desplegable de las prácticas con aquellas que se encuentran asociadas al grupo. En caso de no haber ninguna, el desplegable se muestra vacío.

Una vez seleccionados los valores para la operación pulsaremos en el botón “Desasociar práctica” para completar el proceso. Este proceso no podrá ser ejecutado en caso de que los alumnos integrantes del grupo hayan hecho alguna entrega de la práctica en cuestión, por lo que se abortará el proceso y se notificará al administrador del suceso.



6.1.15 Menú de prácticas

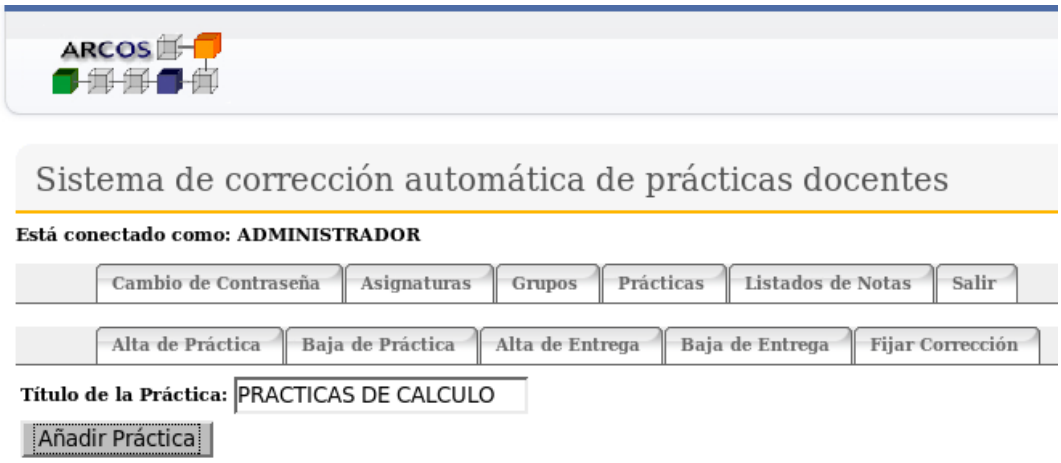
En este menú se permite acceder a las opciones de la aplicación relacionadas con las prácticas docentes del sistema. En él encontramos las siguientes opciones:

- Alta de práctica.
- Baja de práctica.
- Alta de entrega.
- Baja de entrega.
- Fijado de hora de corrección.

6.1.16 Alta de práctica

Esta opción de la aplicación permite dar de alta prácticas en el sistema. Para ello se muestra por pantalla un campo de texto libre en el que el administrador debe introducir la denominación de la práctica.

Una vez introducida la descripción, debe pulsar en el botón “Añadir Práctica” para completar el proceso.



6.1.17 Baja de práctica

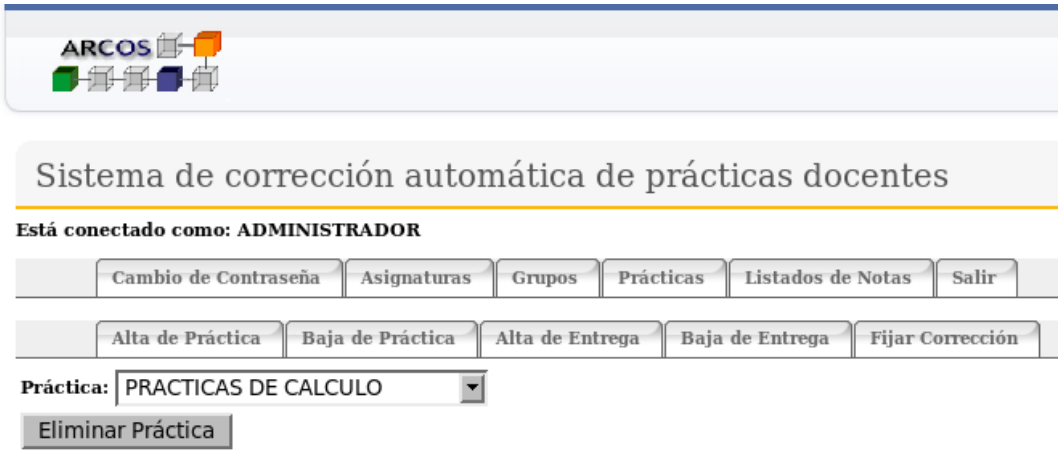
Esta opción ofrece la posibilidad de dar de baja prácticas definidas previamente en el sistema. Para ésto se presenta por pantalla un desplegable con todas las prácticas presentes en el sistema.

Cada práctica se encuentra identificada por su descripción. Para ejecutar el proceso de baja el administrador deberá seleccionar el valor en cuestión y pulsar el botón con la leyenda “Eliminar Práctica”.

Existen dos casos en los que no se podrá dar de baja una práctica, los cuales serán notificados al administrador previo aborto del proceso:

- La práctica tiene asociado un conjunto de entregas. Se deberán eliminar una a una las entregas antes de poder ejecutar la baja (opción “Baja de Entrega” de la barra del menú de prácticas).
- La práctica se encuentra asociada a uno o varios grupos. Se deberán eliminar todas las asociaciones de esa práctica con sus grupos (opción “Quitar Práctica de Grupo” de la

barra del menú de grupos).



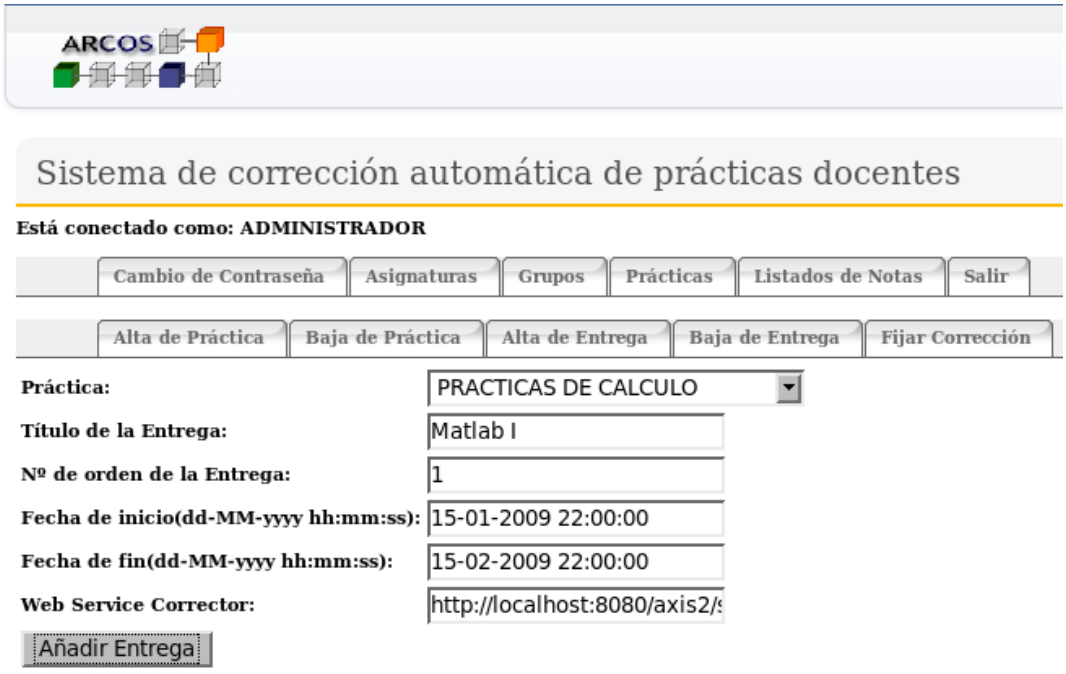
6.1.18 Alta de entrega

Esta opción de la aplicación permite dar de alta una entrega asociada a una práctica determinada. Esto es, que a partir de esta definición los alumnos cuyos grupos tengan asociada la práctica van a poder subir ficheros al sistema y obtener la corrección de los mismos.

Para poder realizar el alta, por una parte se selecciona la práctica en concreto de un desplegable en el que aparecen todas las prácticas presentes en el sistema y una serie de características que van a determinar el comportamiento de la entrega. Dichas características son:

- Título de la entrega: una descripción identificativa de la entrega.
- Número de orden de la entrega: secuencialidad de la entrega dentro de la práctica en concreto.
- Fecha de inicio: fecha a partir de la cual los alumnos van a poder subir ficheros al sistema y obtener su corrección.
- Fecha de fin: fecha hasta la cual los alumnos van a poder subir ficheros al sistema y obtener su corrección.
- Web service corrector: url del servicio web que va a corregir los ficheros subidos por los alumnos.

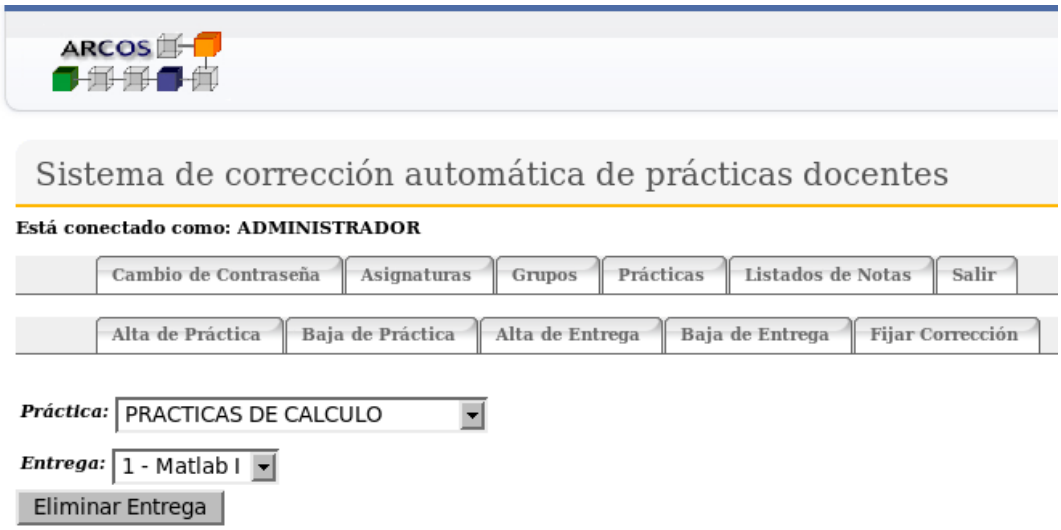
Una vez introducidos todos los datos, el administrador debe pulsar en el botón “Añadir Entrega” para completar el proceso de alta. En caso de que el sistema encuentre un problema relacionado con el formato de los datos o con el mismo proceso de alta, aborta la situación y lo notifica al administrador por pantalla.



6.1.19 Baja de entrega

En este punto de la aplicación el administrador tiene la posibilidad de eliminar una entrega asociada a una práctica en concreto. Para ésto se presentan dos desplegables, uno para las prácticas presentes en el sistema y otro para las entregas asociadas a las mismas.

Cada vez que se seleccione una práctica en concreto se cargarán automáticamente en el otro desplegable las entregas asociadas. Una vez seleccionada la entrega en concreto el administrador debe pulsar el botón “Eliminar Entrega”. Debemos recordar que este proceso solo podrá ser efectivo si no existen alumnos que hayan subido a través de la aplicación ficheros asociados a la entrega, en cuyo caso el sistema aborta el proceso y lo notifica al administrador por pantalla.

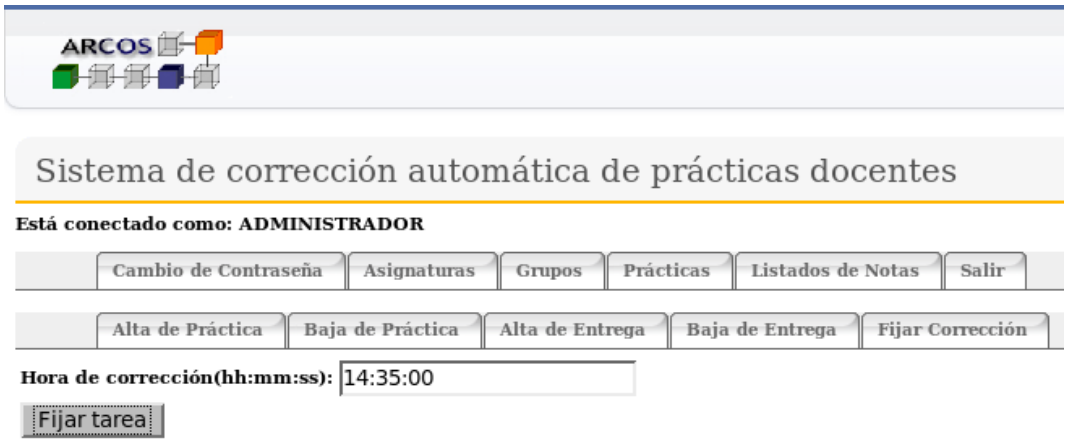


6.1.20 Fijar corrección

En esta parte de la aplicación el administrador puede definir la hora en la que desea que se hagan las correcciones automáticas de todos los ficheros pendientes.

Para ello debe introducir la hora de la corrección en un campo de texto con un formato determinado. Después de ésto solo tendría que pulsar el botón “Fijar Tarea” para completar el proceso.

En caso de que quiera desactivar la corrección automática tendría que dejar en blanco la hora de la corrección y pulsar el botón de “Fijar Tarea”.



6.1.21 Listados de notas

En esta parte de la aplicación el administrador puede obtener informes de las notas del conjunto de alumnos de un grupo sobre una entrega determinada. Para ello se nos presentan tres desplegables por pantalla: uno con los grupos presentes en el sistema, otro con las prácticas y un último con las entregas.

Hay que destacar que cada vez que el administrador selecciona un grupo, se cargarán en el desplegable correspondiente las prácticas asociadas al mismo. En el tercer desplegable se mostrarán las entregas que componen la práctica en cuestión. Una vez seleccionados los parámetros se pulsa el botón “Listado” y se obtiene el listado en formato pdf.

CONSULTA DE NOTAS

Año: 2008 **Grupo:** asa
Asignatura: ARQUITECTURA d **Practica:** PRACTICAS DE ARQUITECTURA
Cuatrimestre: 1 **Entrega:** Programacion de Biestables
Fecha de Inicio: 15/01/09 22:00 **Fecha de Fin:** 30/03/09 22:00

APELLIDOS	NOMBRE	DNI	NIU	NOTA
MORENO PASTOR	LAURA	52984281D	100067298	
ORTEGA SANCHEZ	JAVIER	53429735E	100070937	
SALAS SANCHEZ	SERGIO	53452390E	100073150	
VELA ORTEGA	JAVIER	02270954A	100019024	10,00

6.1.22 Salir

Mediante esta opción se permite al administrador que salga de la aplicación. Para ello únicamente debe pulsar la opción “Salir” y el sistema le dirigirá a la entrada del mismo.

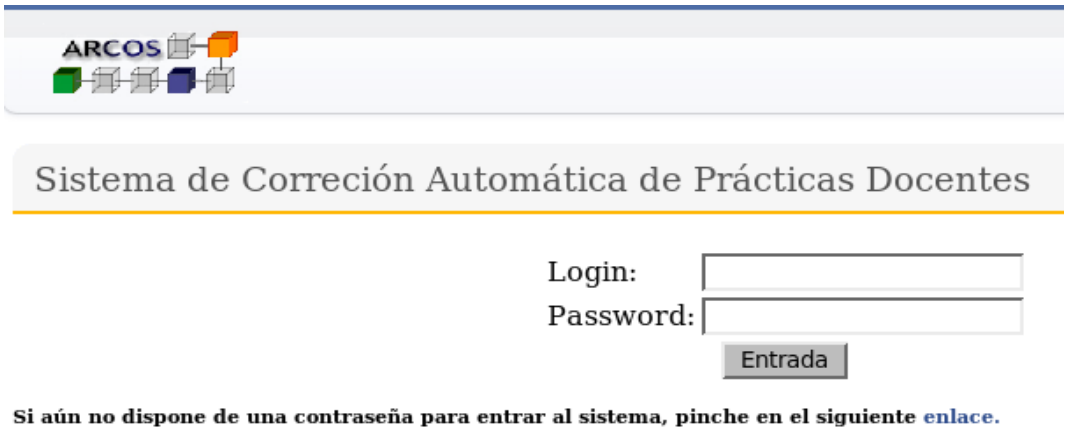
6.2 Documentación orientada al alumno

6.2.1 Acceso a la aplicación

Para acceder a la aplicación el alumno debe introducir la siguiente url en su navegador web:

```
http://<servidor de despliegue>/belane
```

El parámetro <servidor de despliegue> será la dirección web de la máquina donde se despliegue la aplicación.

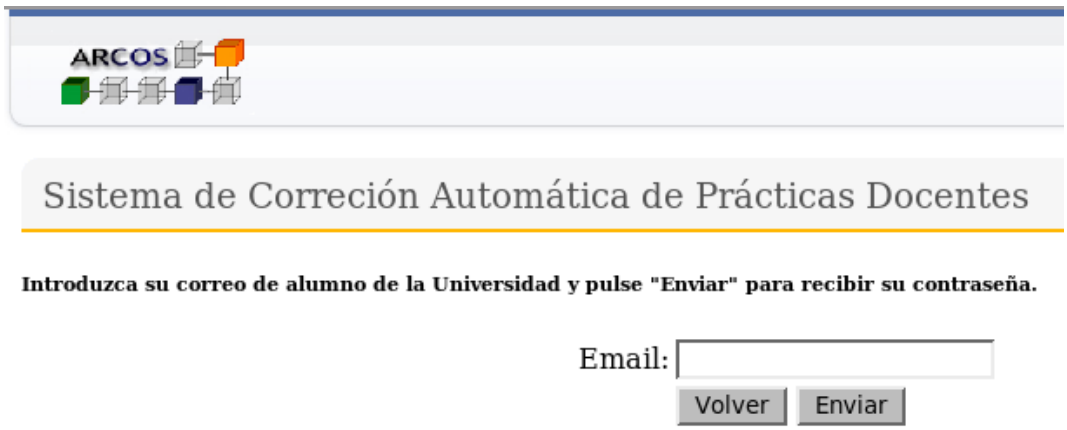


6.2.2 Generación de contraseña

Esta opción del sistema permite generar la contraseña de un alumno, que típicamente la utilizará en dos casos:

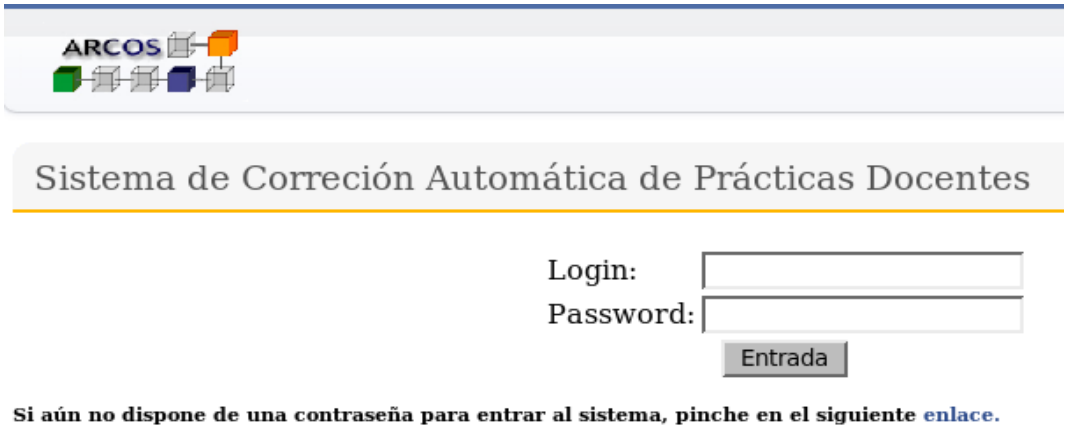
- Antes de entrar por primera vez en el sistema.
- Cuando haya olvidado la contraseña.

En ambos casos el alumno debe introducir en un campo de texto su dirección de correo de alumno asociada y pulsar el botón “Enviar”. Después de esto, el sistema generará una contraseña aleatoria que enviará a la cuenta de correo del alumno. En caso de que exista algún problema, el sistema muestra el mensaje asociado por pantalla.



6.2.3 Entrada al sistema

El alumno accede a la aplicación introduciendo un identificador de usuario y una contraseña asociada. Dicho identificador es el usuario con el que accede al correo de la Universidad, mientras que la contraseña será o bien la que le haya generado automáticamente el sistema o bien la que posteriormente haya establecido el alumno. En caso de no introducir los valores correctos, el sistema le informa de la situación y deniega el acceso a la aplicación.



6.2.4 Barra de menú principal

Esta barra de menú contiene la navegación principal de la aplicación para el alumno, permitiendo acceder a las siguientes opciones:

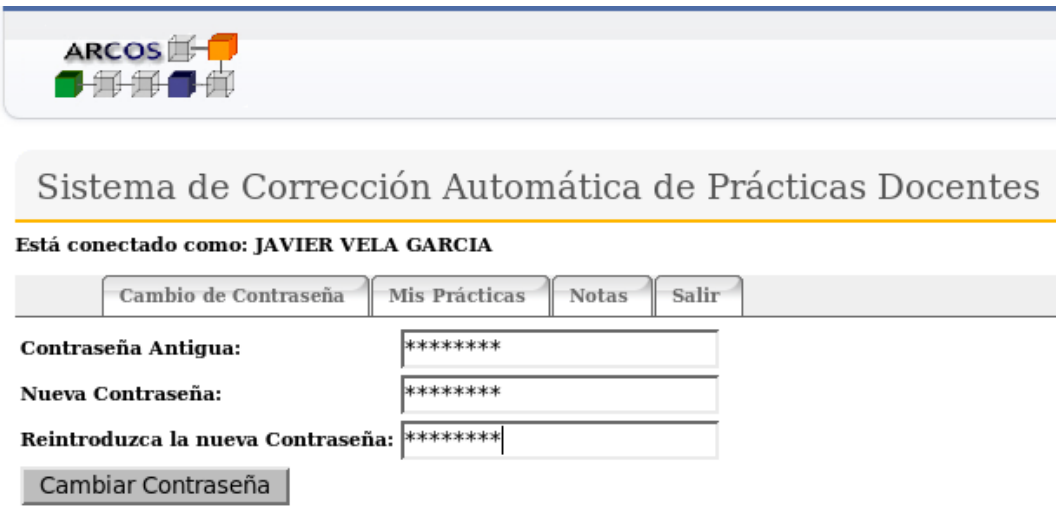
- Cambio de contraseña.
- Mis Prácticas
- Notas
- Salida del sistema

6.2.5 Cambio de contraseña

En esta opción el alumno tiene la posibilidad de cambiar la contraseña de acceso a la aplicación. Para ello se presentan tres campos: uno para introducir la nueva contraseña y otros dos que albergarán el nuevo valor. Cabe señalar que para que se ejecute el cambio deben cumplirse dos condiciones:

- La contraseña actual introducida sea la correcta.
- La contraseña nueva debe ser exactamente igual en los dos campos habilitados para ello.

Una vez introducidos los datos se pulsará el botón con la leyenda “Cambiar Contraseña”.



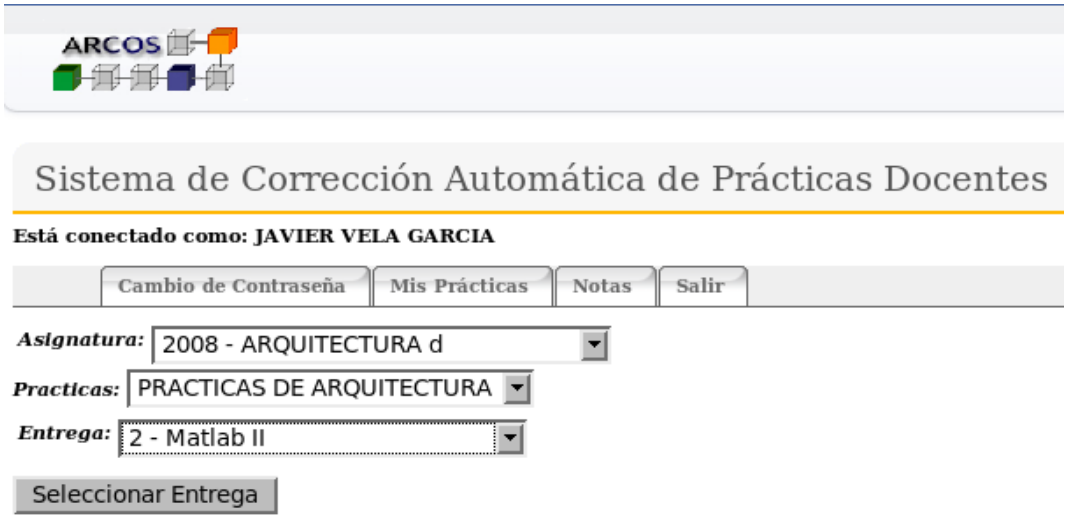
6.2.6 Mis Prácticas

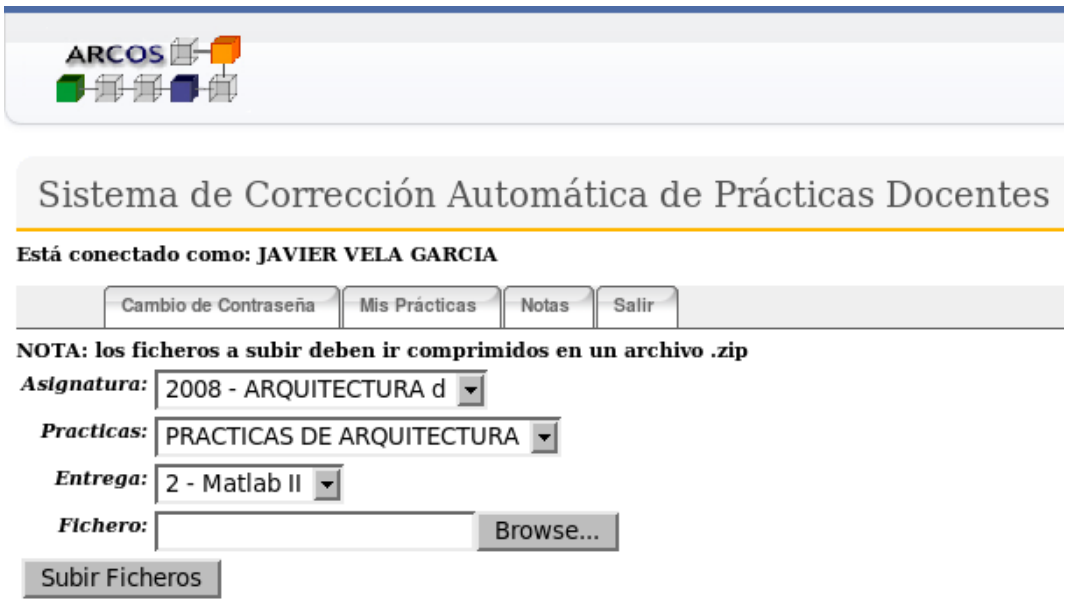
En esta opción de la aplicación el alumno tiene la posibilidad de realizar la entrega del conjunto de ficheros que componen una entrega. Para ésto, en primer lugar se debe seleccionar a través de tres desplegables el grupo, práctica y entrega a la que se quiere asociar los ficheros. En los desplegables únicamente se cargarán los grupos, prácticas y entregas que estén disponibles para el alumno.

Una vez seleccionados los tres parámetros, el alumno debe pulsar el botón “Seleccionar

Entrega" para pasar a la página en la que podrá subir los ficheros. Una vez pulsado el botón, la aplicación le permitirá subir los ficheros. Para ello debe pulsar en el botón "browse" y seleccionar el fichero comprimido .zip que contiene los ficheros que desea asociar a la entrega.

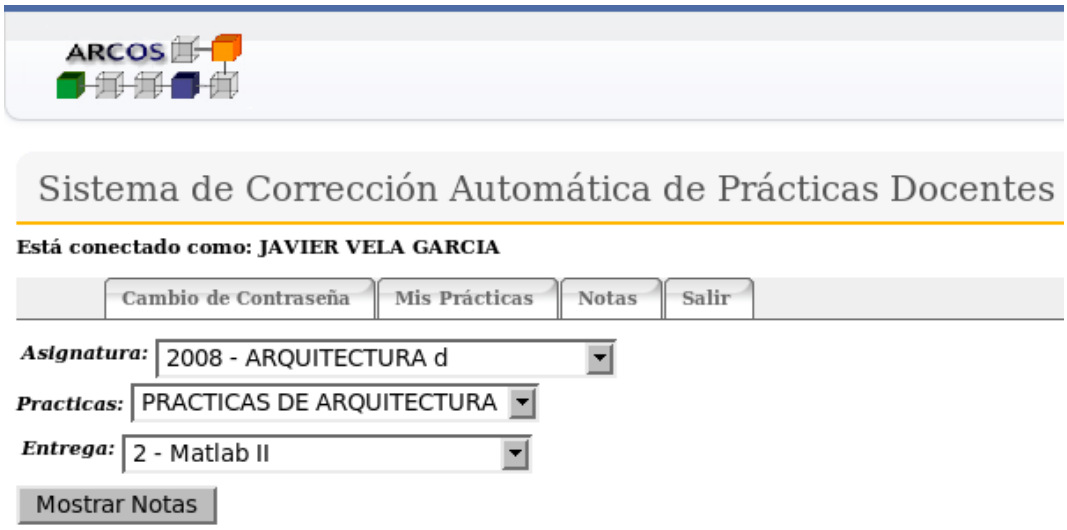
Una vez seleccionado el fichero comprimido, deberá pulsar el botón "Subir Entrega" para entregar los ficheros y que se almacenen en el servidor. En caso de que exista algún problema el sistema mostrará el resultado por pantalla.





6.2.7 Notas

Mediante esta opción el alumno podrá ver las notas de cada una de las entregas. Para ello se debe seleccionar el grupo, práctica y entrega determinados, y pulsar el botón “Mostrar Notas” para obtener un fichero en formato pdf con las notas de las sucesivas entregas realizadas.



CONSULTA DE NOTAS

Año:	2008
Asignatura:	ARQUITECTURA d
Cuatrimestre:	1
Nombre:	JAVIER
Apellidos:	VELA GARCIA
DNI:	52984281D
NIU:	100019024

Practica: PRACTICAS DE ARQUITECTURA
Entrega: Programacion de Biestables

FECHA DE ENTREGA	NOTA
8/02/09 13:58	5,00
12/03/09 18:57	5,00
12/03/09 18:57	10,00

6.2.8 Salir

Mediante esta opción se permite al alumno que salga de la aplicación. Para ello únicamente debe pulsar la opción “Salir” y el sistema le dirigirá a la entrada del mismo.

CAPÍTULO 7:

CONFIGURACIÓN Y PUESTA EN MARCHA

7 Configuración y puesta en marcha

En este apartado se detallarán el conjunto de procedimientos que deben seguirse para desplegar la aplicación en un entorno de producción. En primer lugar hay que señalar que el presente documento de configuración y despliegue es válido para un equipo que tenga un sistema operativo Debian GNU/Linux 4.0 (Etch) con un kernel 2.6.18-4-686.

Es posible obtener una copia gratuita del sistema operativo de los repositorios oficiales del proyecto Debian en http://www.debian.org/distrib/

Es totalmente indispensable que se tenga disponible una cuenta con permisos de administrador.

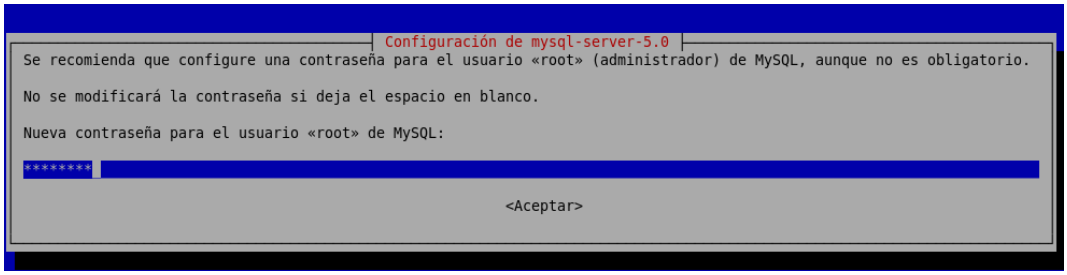
7.1 Base de datos

Como ya se ha indicado en puntos anteriores, se va a utilizar el gestor de base de datos MySQL.

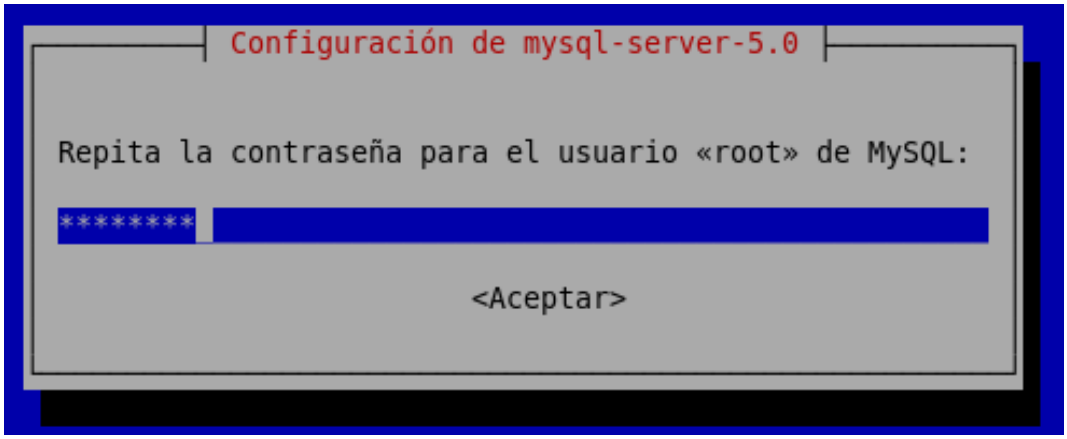
Para instalarlo se debe abrir un terminal de línea de comandos y ejecutar el comando:

```
apt-get install mysql-server
```

Durante el proceso se nos requiere que introduzcamos la contraseña del administrador de mysql.



Se nos requiere de nuevo la contraseña



A continuación nos conectamos a mysql con el comando:

```
mysql -u root -–password
```

y debemos introducir la contraseña establecida en la instalación cuando se nos requiera.

Una vez dentro de mysql creamos la base de datos con la sentencia:

```
create database belane_db;
```

Establecemos el usuario/password con el que vamos a conectarnos, otorgándole los permisos oportunos:

```
GRANT insert,select,update,delete,create,drop,alter ON
belane_db.* to <usuario>@localhost IDENTIFIED BY
'<password>';
```

Hay que señalar que los parámetros <usuario> y <password> corresponden al usuario y contraseña que va a utilizar la aplicación para conectarse a la base de datos.

A continuación ejecutamos el script “script_creacion.sql” contenido en el directorio “scripts” del cd de entrega que crea las tablas de la base de datos (para este caso lo hemos copiado previamente en /tmp):

```
source /tmp/script_creacion.sql
```

Ejecutamos el script “tables_mysql.sql” contenido en directorio “scripts” del cd de entrega que crea las tablas necesarias para implementar la persistencia en el sistema de las correcciones programadas:

```
source /tmp/tables_mysql.sql
```

En este punto ya contaríamos con la base de datos totalmente operativa.

7.2 Máquina Virtual Java

En este apartado se detalla el procedimiento a seguir para instalar la máquina virtual Java necesaria sobre la que va a funcionar el servidor de aplicaciones. Para la realización de este proyecto se ha utilizado el Java Development Kit (JDK) versión 1.6.0_07.

Puede obtenerse la versión actualizada de la web de Sun Microsystems: http://java.sun.com/javase/downloads/index.jsp

Una vez obtenido el paquete necesario se copia en /usr/local, se le otorgan permisos de ejecución y se ejecuta el archivo:

```
chmod 755 jdk-6u7-linux-i586.bin
./jdk-6u7-linux-i586.bin
```

Se aceptan las condiciones de la licencia y acto seguido comienza el proceso de descompresión del archivo.

Una vez terminado el proceso creamos un enlace simbólico con nombre jdk apuntando al directorio que se nos acaba de crear:

```
ln -s /usr/local/jdk1.6.0_07/
/usr/local/jdk
```

7.3 Servidor de aplicaciones

Como servidor de aplicaciones se ha utilizado la versión 6.0.18 de Apache Tomcat, que puede obtenerse gratuitamente de la página web oficial del proyecto: http://tomcat.apache.org

Se descarga el archivo comprimido apache-tomcat-6.0.18.tar.gz en el directorio /usr/local

A continuación y desde e /usr/local se descomprime con el comando:

```
tar -xvzf  apache-tomcat-6.0.18.tar.gz
```

El resultado de la ejecución crea el directorio apache-tomcat-6.0.18 en /usr/local . Para mayor comodidad de administración, crearemos un enlace simbólico ejecutando:

```
ln -s /usr/local/apache-tomcat-6.0.18 /usr/local/tomcat
```

Editamos los fichero /usr/local/tomcat/bin/startup.sh y /usr/local/tomcat/bin/shutdown.sh y añadimos la siguiente línea para indicarle al servidor de aplicaciones donde se ubica la máquina virtual Java que debe utilizar:

```
export JAVA_HOME=/usr/local/jdk
```

Para iniciar la ejecución del servidor de aplicaciones ejecutaremos:

```
/usr/local/tomcat/bin/startup.sh
```

Para parar la ejecución del servidor de aplicaciones se ejecuta:

```
/usr/local/tomcat/bin/shutdown.sh
```

7.4 Paso al entorno de explotación

En este apartado se define los procedimientos necesarios para desplegar y configurar la aplicación en un entorno de explotación.

7.4.1 Despliegue de la aplicación

En primer lugar debemos copiar la librería “mysql-connector-java-5.1.6-bin.jar” contenida en el directorio “lib” del cd de entrega al directorio /usr/local/tomcat/lib

Esta librería, obtenida de http://www.mysql.com, permitirá la conexión a la base de datos desde la aplicación desplegada en el servidor de aplicaciones.

A continuación, debemos copiar el archivo “belane.war” del directorio “app” del cd de entrega al directorio /usr/local/tomcat/webapps.

Después de todo esto debemos ejecutar el siguiente comando para arrancar el servidor de aplicaciones:

```
/usr/local/tomcat/bin/startup.sh
```

Una vez alcanzado este punto ya tenemos la aplicación desplegada en un entorno de producción.

7.4.2 Configuración

En esta sección se detallan todas las opciones y parámetros sujetos a modificación de la aplicación

Fichero context.xml

Después de desplegar la aplicación se sitúa en /usr/local/tomcat/webapps/belane/META-INF y en él se definen los parámetros relativos a la conexión a la base de datos desde la aplicación.

Su contenido es:

```
<?xml version="1.0" encoding="UTF-8"?>
<Context>
    <Resource name="jdbc/belane_db"
   auth="Container"
   type="javax.sql.DataSource"
```

```
        username="<usuario>"
        password="<password>"
        driverClassName="com.mysql.jdbc.Driver"
        url="jdbc:mysql://localhost:3306/belane_db"
        maxActive="8" maxIdle="4"/>
         <WatchedResource>WEB-INF/web.xml</WatchedResource>
         <WatchedResource>META-INF/context.xml</WatchedResource>
    </Context>
```

Hay que señalar que "usuario" y "password" contienen los valores que hemos definido en el apartado 7.1 del presente documento.

Fichero quartz.properties

En este fichero se define la configuración de la librería quartz, componente en el que se apoya la aplicación para gestionar la programación de las correcciones automáticas. Tiene el siguiente contenido:

```
org.quartz.scheduler.instanceName = DefaultQuartzScheduler
org.quartz.scheduler.rmi.export = false
org.quartz.scheduler.rmi.proxy = false
org.quartz.scheduler.wrapJobExecutionInUserTransaction = false
org.quartz.threadPool.class = org.quartz.simpl.SimpleThreadPool
org.quartz.threadPool.threadCount = 10
org.quartz.threadPool.threadPriority = 5
org.quartz.threadPool.threadsInheritContextClassLoaderOfInitializingThread = true
org.quartz.jobStore.misfireThreshold = 60000
org.quartz.jobStore.class = org.quartz.impl.jdbcjobstore.JobStoreTX
org.quartz.jobStore.tablePrefix = QRTZ_
org.quartz.jobStore.driverDelegateClass = org.quartz.impl.jdbcjobstore.StdJDBCDelegate
org.quartz.jobStore.dataSource = qzDS
org.quartz.dataSource.qzDS.driver = com.mysql.jdbc.Driver
org.quartz.dataSource.qzDS.URL = jdbc:mysql://localhost:3306/belane_db
```

org.quartz.dataSource.qzDS.user = <usuario>

org.quartz.dataSource.qzDS.password = <password>

org.quartz.dataSource.qzDS.maxConnections = 60

org.quartz.plugin.jobInitializer.class = org.quartz.plugins.xml.JobInitializationPlugin

org.quartz.plugin.jobInitializer.overWriteExistingJobs = true

org.quartz.plugin.jobInitializer.failOnFileNotFound = true

org.quartz.plugin.jobInitializer.validating=false

Al igual que en el fichero context.xml, los valores "usuario" y "password" deben coincidir con los valores definidos en el apartado 7.1. También debemos señalar que los valores de los parámetros org.quartz.dataSource.qzDS.driver y org.quartz.dataSource.qzDS.URL coinciden con los valores de "driverClassName" y "url" del fichero "context.xml", ya que definen respectivamente la clase utilizada para acceder a la base de datos y la url a través de la que es accesible.

Fichero belane.properties

En este fichero se definen los parámetros propios del funcionamiento de la aplicación. Su contenido es el siguiente:

DATA_SOURCE=1

ADMINISTRATOR=admin

ROOTPATH=<path>

SMTP_SERVER=<smtp>

MAIL_ADMIN=<mail admin>

ANO_ACADEMICO=2008

SUBJECT_MAIL_PASSWORD=Datos para el sistema de corrección de practicas docentes

BODY_MAIL_PASSWORD=Sistema de Corrección de Prácticas Docentes:\n Utilice los siguientes datos para entrar en el sistema:\n Usuario:<niu>\n Contraseña:<password>

SUBJECT_MAIL_ENTREGA:Nota provisional de la entrega de

ENCABEZADO_BODY_MAIL_ENTREGA:El resultado de la compilación de los archivos es:\n

ERROR_WEBSERVICE:Existen errores en la compilación.\n

EXITO_WEBSERVICE:Archivo compilado.\N

NOTA_BODY_MAIL_ENTREGA:La nota global de la entrega es:

CORRECTOR=1

OPERACION_COMPILACION= getCompilacion

OPERACION_NOTA = getNota

A continuación se detallan los parámetros uno por uno:

DATA_SOURCE

Define la implementación concreta que se va a utilizar para los objetos de acceso a datos.

ROOTPATH

Define el directorio raíz del bajo el que se almacena la estructura de ficheros que contiene los distintos grupos, prácticas y entregas realizadas por los alumnos.

SMTP_SERVER

Determina el servidor smtp que utiliza la aplicación para enviar las distintas notificaciones a los alumnos por correo electrónico.

MAIL_ADMIN

Define la cuenta de correo que va enviar por correo electrónico las notificaciones a los alumnos.

ANO_ACADEMICO

Representa el año académico en el que se engloban los distintos grupos. Cabe destacar que para un año académico de la forma XXXX-YYYY el valor que debemos introducir es XXXX (por ejemplo 2008 para un año académico 2008-2009).

SUBJECT_MAIL_PASSWORD

Define el asunto del correo que se envía a los alumnos con la reinicialización de la contraseña.

BODY_MAIL_PASSWORD

En este parámetro configuraremos el texto que aparecerá en el correo de reinicialización de la contraseña del alumno.

SUBJECT_MAIL_ENTREGA

Permite definir el asunto del correo que reciben los alumnos cuando el sistema corrige una

entrega.

ENCABEZADO_BODY_MAIL_ENTREGA

Define el encabezado del cuerpo del correo que reciben los alumnos cuando el sistema corrige una entrega.

ERROR_WEBSERVICE

Mensaje de error que aparecerá en la notificación de la corrección por cada uno de los ficheros que no sea correcto.

EXITO_WEBSERVICE

Mensaje asociado a cada uno de los ficheros corregidos correctamente en el correo de notificación de corrección.

NOTA_BODY_MAIL_ENTREGA

Define el texto que aparece antes de la nota global de la entrega en el correo de notificación de corrección al alumno.

CORRECTOR

Este parámetro define la factoría concreta de objetos de acceso a datos que va a implementar el corrector en el sistema.

OPERACION_COMPILACION

Define la operación que da el resultado de la compilación de un archivo y que se va a utilizar en la llamada al servicio web.

OPERACION_NOTA

Operación que va a proporcionar la nota de un archivo en concreto y es ofrecida por el web service.

CAPÍTULO 8:

GESTIÓN DEL PROYECTO

8 Gestión del proyecto

En este capítulo se detallan los aspectos del proyecto relacionados con la propia gestión del mismo.

8.1 Planificación

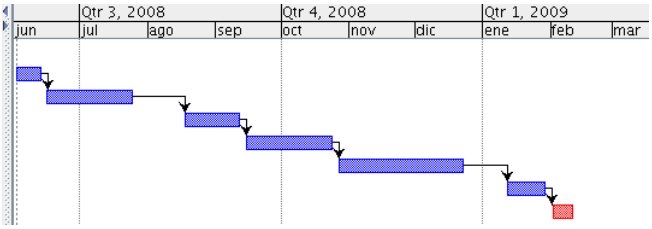
Para representar la ejecución del proyecto a lo largo del tiempo se han empleado diagramas de Gantt. Para ello se ha utilizado la herramienta OpenProj, que es un elemento software capaz de representar eficazmente el desarrollo de una aplicación a lo largo del tiempo.

Para la elaboración del proyecto se ha utilizado el modelo de desarrollo en cascada, que típicamente está compuesta de las siguientes fases:

- Evaluación de necesidades
- Análisis de requisitos
- Diseño del sistema
- Diseño del programa
- Implementación
- Pruebas
- Implantación
- Mantenimiento

A continuación se presenta el diagrama Gantt que refleja las distintas fases del ciclo de vida de la aplicación frente al tiempo. Hay que señalar que se ha prescindido de la fase de mantenimiento por considerar que es externa al ámbito de este proyecto. Asimismo se ha añadido una fase al inicio del calendario, “Evaluación de necesidades”, en la que se han determinado los recursos a emplear en todo el proceso. Hay que señalar que se suponen 6 horas para cada día de trabajo estimado.

	Nombre	Duración	Inicio	Terminado
	⊟	183 days	2/06/08 8:00	11/02/09 17:00
1	Evaluacion de necesidades	10 days	2/06/08 8:00	13/06/08 17:00
2	Analisis de requisitos	30 days	16/06/08 8:00	25/07/08 17:00
3	Diseño del sistema	20 days	18/08/08 8:00	12/09/08 17:00
4	Diseño del programa	30 days	15/09/08 8:00	24/10/08 17:00
5	Implementacion	42 days	27/10/08 8:00	23/12/08 17:00
6	Pruebas	14 days	12/01/09 8:00	29/01/09 17:00
7	Implantacion	8 days?	1/02/09 8:00	11/02/09 17:00



En el siguiente cuadro se presentan las distintas fases junto con el número de horas que ha comprendido cada una:

	Nombre	Trabajo	Duración	Inicio	Terminado
1	Evaluacion de necesidades	60 horas	10 days	2/06/08 8:00	13/06/08 17:00
2	Analisis de requisitos	180 horas	30 days	16/06/08 8:00	25/07/08 17:00
3	Diseño del sistema	120 horas	20 days	18/08/08 8:00	12/09/08 17:00
4	Diseño del programa	180 horas	30 days	15/09/08 8:00	24/10/08 17:00
5	Implementacion	252 horas	42 days	27/10/08 8:00	23/12/08 17:00
6	Pruebas	84 horas	14 days	12/01/09 8:00	29/01/09 17:00
7	Implantacion	48 horas	8 days?	1/02/09 8:00	11/02/09 17:00

Total de horas: 924

8.2 Recursos utilizados

En este apartado se detalla el conjunto de recursos utilizados. Para cada uno se detalla el coste económico.

8.2.1 Herramientas software

Herramienta	*Descripción*	*Coste (€)*
Ubuntu 8.04 (Hardy Heron)	Sistema operativo utilizado en la máquina en la que se realizó el proyecto.	Gratuito
Debian GNU/Linux 4.0 (Etch)	Sistema operativo utilizado en la máquina en la que se han realizado pruebas de despliegue y configuración.	Gratuito
Apache Tomcat 6.0.18	Servidor de aplicaciones utilizado para contener la aplicación.	Gratuito
Eclipse 3.4.0	Entorno de desarrollo integrado empleado para codificar la aplicación.	Gratuito
Java Development Kit 1.6.0_07	Herramienta de desarrollo Java que contiene el compilador y la máquina virtual Java.	Gratuito

Herramienta	***Descripción***	***Coste (€)***
struts 2.0.14	Framework que implementa el patrón Modelo Vista Controlador (MVC).	Gratuito
Apache axis2 1.4.1	Implementación de SOAP utilizada junto al contenedor de servlets Tomcat para contener los servicios web de prueba.	Gratuito
iText 2.1.4	Librería para generar archivos pdf.	Gratuito
Jasperreports 3.1.3	Librería para generar informes.	Gratuito
Mysql-connector-java 5.1.6	Driver para conectar la aplicación con el sistema gestor de bases de datos MySQL.	Gratuito
Pwgen	Librería para generar contraseñas aleatorias.	Gratuito
Quartz 1.6.4	Librería para poder establecer procesos batch desde la aplicación.	Gratuito
Serena OpenProj 1.4	Aplicación empleada para la gestión del proyecto.	Gratuito
MySQL 5.0	Sistema de gestión de bases de datos relacionales.	Gratuito
DBDesigner 4.0.5.4	Herramienta de modelado de bases de datos relacionales.	Gratuito
Ireport 3.0.0	Herramienta de diseño de informes.	Gratuito
Umbrello 1.5.8	Herramienta de modelado de diagramas UML.	Gratuito
DIA 0.96.1	Herramienta de diseño de diagramas.	Gratuito
OpenOffice 2.2.4	Suite ofimática.	Gratuito
TOTAL COSTE SOFTWARE		***0,00 €***

8.2.2 Herramientas hardware

Herramienta	*Características*	*Coste (€)*
Ordenador portátil Samsung P30	• Procesador: Intel Pentium M 1400 Mhz • Memoria RAM : 512 MB • Disco Duro: 40 GB • Tarjeta Gráfica: Radeon Mobility 9200 • Chipset: 855GM, • Pantalla: 15.1 pulgadas • Grabador de DVD • Tarjeta de red inalámbrica	1.300 ,00
COSTE TOTAL HARDWARE		***1.300,00 €***

8.2.3 Servicios

Servicio	*Descripción*	*Coste (€)*
Conexión	ADSL 6Mb durante nueve meses, 44 euros/mes.	396,00
COSTE TOTAL SERVICIOS		***396,00€***

8.2.4 Personal

En la siguiente tabla se detallan los costes de personal. Para ello se detallan los perfiles profesionales empleados en el proyecto, su coste por hora y el número total de horas parte del total especificado en el punto 8.1. Los salarios de cada uno de los perfiles se han obtenido del portal de empleo infojobs (http://salarios.infojobs.net)

Perfil	*Coste/Hora (€)*	*Nº horas*	*Coste (€)*
Jefe de proyecto	20	200	4.000,00
Analista	15	340	5.100,00
Programador	12	336	4.032,00
Técnico de sistemas	10	48	480,00
COSTE TOTAL DE PERSONAL			***13.612,00 €***

8.3 Resumen de costes totales

En la siguiente tabla se indican los costes totales imputables al desarrollo del proyecto:

Concepto	*Coste*
Software	0,00 €
Hardware	1.300,00 €
Servicios	396,00 €
Personal	13.612,00 €
Coste total (Sin I.V.A)	***15.308,00 €***

8.4 Presupuesto para el cliente

Para establecer un presupuesto para un cliente potencial se debe asegurar de manera razonada la rentabilidad económica del proyecto. Una vez calculados los costes totales sin IVA en el apartado anterior, se procede a calcular el beneficio, que será fijado en un 20% del coste total.

En la siguiente tabla se detallan todos los conceptos significativos:

Concepto	***Valor (€)***
Coste del proyecto (sin I.V.A)	15.308,00
Beneficio del proyecto (20% del total de los costes)	3.061,60
Precio de realización del proyecto (sin I.V.A.)	18.369,60
I.V.A. (16%)	2.939,14
Precio para el cliente	***21.308,74 €***

CAPÍTULO 9:

CONCLUSIONES

9 Conclusiones

El presente trabajo me ha servido para tomar conciencia de la gran importancia del proceso de pruebas en el ciclo del desarrollo software. Considero que la fase de pruebas y correcciones del código es una etapa fundamental que permite generar aplicaciones de mayor calidad. Concretamente me ha parecido muy interesante el componente pedagógico del proyecto, ya que la aplicación constituye una herramienta totalmente válida para que los alumnos generen unos mejores ejercicios y puedan evaluar sus resultados a lo largo de los periodos de validez de las prácticas.

Por otra parte, el hecho de que este proyecto tenga una aplicación práctica en el ámbito académico de la Universidad ha supuesto una gran motivación para su realización.

En cuanto al ciclo de vida de la aplicación, me gustaría recalcar la gran importancia de hacer un buen análisis de la aplicación, ya que considero que el establecimiento de requisitos como el modelado del dominio en el que se define el sistema a tratar determina la totalidad del proceso de desarrollo software.

También se ha podido comprobar la utilidad de utilizar el patrón de diseño Modelo-Vista -Controlador (MVC). Creo que se trata de la aproximación idónea para acometer todo proyecto que implique el diseño e implementación de una aplicación orientada a la web. He podido ver claramente como el empleo de este patrón permite generar aplicaciones más legibles, mejor estructuradas, más fáciles de mantener y más modulares.

En el sentido del punto anterior, creo que el empleo de la plataforma Java EE combinándola con Struts 2 y páginas jsp es una solución totalmente satisfactoria para generar aplicaciones de distinta índole.

Respecto al conjunto de tecnologías empleadas, considero que es un acierto el intentar favorecer soluciones de código abierto. Esto implica múltiples ventajas como el respaldo de grandes comunidades de desarrolladores y usuarios, gran ahorro de costes, mayor transparencia y la

posibilidad de llegar tan lejos como se quiera gracias a la disponibilidad del código.

Por último quería referirme a la tecnología de los servicios web. Considero que es algo que debe tenerse muy en cuenta para futuras desarrollos. Si bien no resulta un concepto del todo novedoso (invocación de métodos remotos alojados en otro sistema), es una buena solución para lograr la interoperabilidad entre distintos sistema utilizando un intercambio estándar de datos.

CAPÍTULO 10:

LÍNEAS FUTURAS

10 Líneas futuras

En este apartado se exponen los trabajos futuros sobre el sistema desarrollado. Cualquier aplicación o trabajo siempre es susceptible de mejora, por lo que se establecen las siguientes propuestas:

- Un punto importante en que se considera que se debe ahondar es la parte del corrector de ejercicios. Por una parte se propone la posibilidad de diseñar e implementar distintos correctores. El abanico de posibilidades es inmenso, ya que se pueden enfocar a distintos lenguajes de programación. Siguiendo esta línea, sería bastante interesante profundizar en el estudio y desarrollo de los servicios web, ya que constituye una solución tecnológica clave para el futuro de las aplicaciones web.

- En la misma línea que la anterior propuesta, otro posible punto de valor añadido al presente trabajo es realizar un estudio de las distintas tecnologías e implementaciones de los servicios web para dilucidar cual es la idónea para emplear con el sistema actual.

- Otra posible aproximación puede ser el hacer trabajar la aplicación con la base de datos de gestión académica de la Universidad. Esto permitiría a la aplicación trabajar directamente y en tiempo real con la información referente a asignaturas, grupos de clase y alumnos matriculados en ellos sin necesidad de cargar la composición de los mismos con un fichero. Esto no implicaría un gran conjunto de cambios, ya que se trataría de cambiar la implementación de las distintas entidades de datos con las que trabaja la aplicación para que se instanciaran a partir de otra fuente de datos. Esto es posible gracias a la utilización del patrón de diseño Data Access Object (DAO) , por lo que los cambios en la lógica de negocio de la aplicación serían mínimos.

- Por último se propone implementar una gestión más eficiente de la configuración de la aplicación. Actualmente los distintos parámetros de la aplicación se alojan en un fichero de texto que debe ser editado de forma externa al sistema. Esta mejora consistiría en

llevar la gestión de estos parámetros directamente a la aplicación y evaluar la idoneidad de seguir almacenando estos parámetros en un fichero o llevarlo al plano de la base de datos.

CAPÍTULO 11:

BIBLIOGRAFÍA Y REFERENCIAS WEB

11 Bibliografía y referencias web

"Expert One-on-One J2EE Design and Development"

Rod Johnson

Wrox, 2002

"Jsp Tag Libraries"

Gal Shchor, Adam Chace, Magnus Rydin.

Manning, 2002

"Java 2: Lenguaje y aplicaciones"

Francisco Javier Ceballos Sierra

Ra-Ma, 2006

"UML Bible"

Tom Pender

Wiley,2003

"Struts 2 in Action"

Donald Brown, Chad Michael Davis, Scott Stanlick

Manning, 2008

"Apache Tomcat Bible"

Donald Brown, Chad Michael Davis, Scott Stanlick

Manning, 2008

"The Definitive Guide To JasperReports"

Teodor Danciu,Lucian Chirita

Apress, 2007

"Patrones de diseño: elementos software orientados a objetos reutilizable"

Erich Gamma

Addison Wesley, 2002

"UML y patrones: Introducción al análisis y diseño orientado a objetos y al proceso unificado"

Craig Larman

Prentice-Hall, 2002

Interfaz de programación de aplicaciones Java:

http://java.sun.com/javaee/5/docs/api

Sitio web enfocado a las pruebas de software:

http://www.testingbrain.com/

Versión en inglés del proyecto Wikipedia:

http://en.wikipedia.org

12 Anexo: Acrónimos

AJAX: Asynchronous Javascript And Xml

API: Application Programing Interface

ASP: Active Server Pages

CD: Compact Disc

CGI: Common Gateway Interface

CU: Caso de Uso

DAO: Data Access Object

DNI: Documento Nacional de Identidad

DTO: Data Transfer Object

E: Excepción

EJB: Enterprise Java Bean

GNU: GNU's Not Unix

GPL: General Public License

HTML: Hypertext Markup Language

HTTP: HyperText Transfer Protocol

IIS: Internet Information Services

IVA: Impuesto al Valor Agregado

JAR: Java ARchive

JDK: Java Development Kit

JEE: Java Enterprise Edition

JSP: Java Server Pages

JSTL: Java server pages Standard Tag Libary

NIU: Número de Identificación de Usuario

MD5: Message-Digest algorithm 5

MVC: Modelo Vista Controlador

ONGL: Object Graph Notation Language

PDF: Portable Document Format

POJO: Plain Old Java Object

RI: Requisito de Interfaz

RF: Requisito Funcional

SGBD: Sistema Gestor de Bases de Datos

SOAP: Simple Object Access Protocol

SQL: Structured Query Language

UDDI: Universal Description Discovery and Integration

UML: Unified Modeling Language

URI: Uniform Resource Identifier

WSDL: Web Services Description Language

XML: eXtensible Markup Language